Burroughs Corporation 🅑

COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

BASIC S-LANGUAGE

PRODUCT SPECIFICATION

3 0 MAY 1973

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED ADDED DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
A	2-20-73	Original Issue	WFK	
B	5-24-73	Sec. 1.1.1 Reversed order of items 2 & 3.	WFK	
		Sec 2.4 Deleted requirement that value zero have exponent of zero.		
		Sec 3.5 Fix: Clarified		
		Sec 3.8 & 3.9 AND and OR: Deleted		
		Sec 3.18 & 3.19 LA and LV changed OP codes.		
		Sec 3.20 ERS: Stated subscript is out of bounds if it is equal to the bounds value.		
		Sec 3.23 & 3.24 FOR and NEXT: Corrected order in which parameters are stored.		
		Sec 3.28 & 3.29 CALL and NTR: Added statement that hardware stacks are stored in memory.		
		Sec 3.29 RTN: Corrected to state that stack is cut back to RCW before STACK. CUT.VALUE is used.		
		Sec 3.30 COMM: Deleted FIX bit. Address is always base relative in non-overlayable area and does not need conversion.		

TABLE OF CONTENTS

Table of Contents (Cont'd.)

1.0   GENERAL DESCRIPTION

All BASIC S-Language programs have associated with them a Base Regis-
ter and a Limit Register. The area between the Base and Limit is
used as a data space. This data space includes a non-overlayable area
which contains the STACK and the DATA TABLE. It also includes an over-
layable area for data arrays.

Immediately above the data area defined by the program's Limit Regis-
ter is an area known as the program's Run Structure Nucleus (RS.
NUCLEUS). This RS. NUCLEUS contains the information necessary for
the MCP and interpreter to execute the program.

All other object program modules such as code segments, code and data
dictionaries, file information blocks and buffers are maintained out-
side the base and limit bounds by the MCP.

BASE

```
                  ┌─────────────────────────┐    ┌─ ─ ─ ─ ─ ─ ─ ─ ─┐
                  │                         │                    ↑
                  │         STACK           │                    │
                  │                         │                    │
                  ├─────────────────────────┤                    │
                  │       DATA TABLE        │                    │
                  │          OF             │         NON-OVERLAYABLE
                  │   SIMPLE  VARIABLES      │                    │
                  │          AND            │                    │
                  │   ARRAY  DESCRIPTORS     │                    │
                  ├─────────────────────────┤ ─ ─ ─ ─ ─ ─ ─ ─ ─ ─⋁⋀─
                  │                         │                    │
                  │        ARRAY            │                    │
                  │        AREA             │          OVERLAYABLE
                  │                         │                    │
         LIMIT    ├─────────────────────────┤ ───────────── ⋁  EXTENDABLE AT BOJ
                  │                         │
                  │   RUN  STRUCTURE        │
                  │      NUCLEUS            │
                  │                         │
                  │                         │
                  └─────────────────────────┘

                  ┌─────────────────────────┐
                  │        S-CODE           │
                  └─────────────────────────┘
```
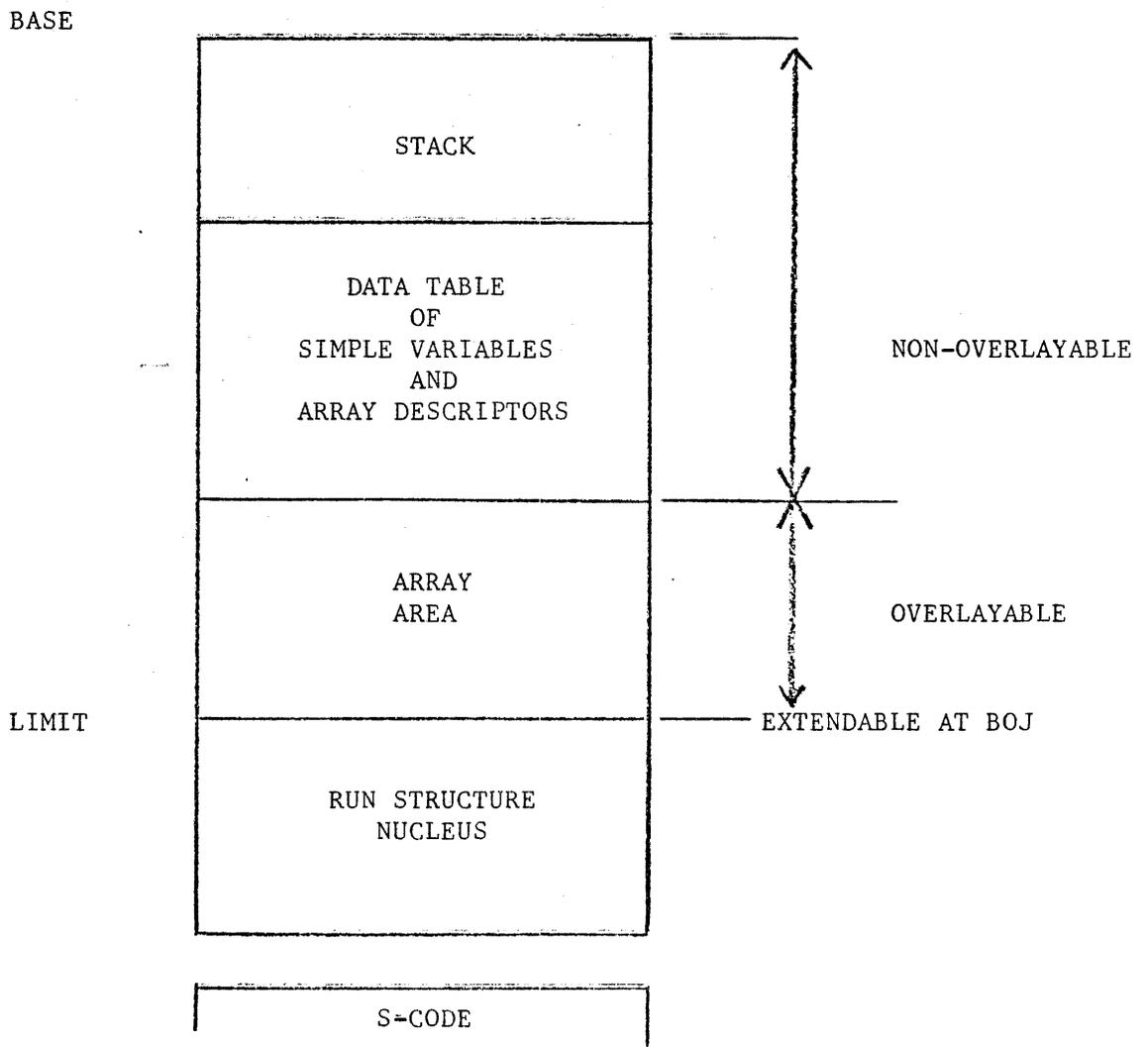
Basic Object Program Memory Layout.

FIGURE 1

1.1   RS.NUCLEUS Format

The RS.NUCLEUS is located immediately above the data defined by the
program's limit register. The RS.NUCLEUS contains information neces-
sary for the MCP and the interpreter to execute the program. A pro-
grammatic description of the RS.NUCLEUS is given in Figure 2.

```
        DECLARE RS.NUCLEUS TEMPLATE BIT (RS.N.SIZE);%
        DEFINE  RS.N.DECLARATION AS #%
DECLARE 01 DUMMY REMAPS RS.NUCLEUS,%
            02 RS.COMMUNICATE.MSG.PTR      BIT (48)        ,%
            02 RS.REPLY                    BIT (48)        ,%
            02 RS.COMMUNICATE.LR           ADDRESS,
            02 RS.REINSTATE.MSG.PTR        BIT (48)        ,%
            02 RS.MY.BASE                  ADDRESS         ,%
            02 RS.MY.LIMIT                 ADDRESS         ,%
            02 RS.MCP.BIT                  BOOLEAN,
            02 RS.NIP                      BIT (32)        ,%
            02 RS.SEG.DIC.PTR              BIT (24)        ,%
            02 RS.DATA.DIC                 ADDRESS         ,%
            02 RS.INTERP.ID                BIT (5)         ,%
            02 RS.INTRINSICS.LOC           BIT (10),
            02 RS.M.MACHINE                BIT(S.PAD.SIZE),%
            02 RS.FIB.DIC                  ADDRESS         ,%
            02 RS.PRIORITY.INTEGER         BIT (4),
            02 RS.PRIORITY.FRACTION        BIT (4),
            02 RS.LOG.PTR                  DSK.ADR.        ,%
            02 RS.WAIT.HDWR                BIT (6)         ,%
            02 RS.STATUS.SCRATCHPAD        WORD            ,%
            02 RS.NXT.RQ                   ADDRESS         ,%
            02 RS.LAST.OVLY                ADDRESS,
            02 RS.DATA.OVERLAYS            ADDRESS,
            02 RS.LAST.LINK                ADDRESS,
            02 RS.FIRST.AVAIL              ADDRESS,
            02 RS.OVLY.DISK.BASE           DSK.ADR,
            02 RS.OVLY.DISK.PTR            WORD,
            02 RS.OVLY.DISK.SIZE           WORD,
            02 RS.MIX.NMBR                 BIT (8),     % INDEX TO "MIX" TABLE
            02 RS.NUMBER.FILES             BIT (8)      % 255 FILES ALLOWED
            02 RS.TRACK                    ADDRESS,     % TRACE INFO ADDRESS
            02 RS.OVLY.DESC                BIT (DESCRIPTOR.SIZE);
        #;%
```

FIGURE 2.   RUN STRUCTURE MAINTAINED BY MCP

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 3
BASIC S-LANGUAGE
P.S. #2210 0135

1.1.1   Program Parameters

The following 24 bit parameters are included in the RS.NUCLEUS.
These parameters, in the order given, are placed in the scratchpad
disk segment of the object program by the compiler and are main-
tained in the RS.M.MACHINE area of the RS.NUCLEUS by the Basic In-
terpreter which updates them whenever control is transferred to the
MCP.

1)   DATA.TABLE.BASE

This is the Base relative bit address
of the base of the data table. It is
also the original stack size in bits.

2)   STACK.LENGTH

This is maintained as the length of
the stack not currently in use.

3)   STACK.ADDRESS

This is maintained as the base rela-
tive address of the next available
stack word. It is initially zero.

4)   RETURN.CONTROL.REGISTER

This contains the current contents of
the return control register which lo-
cates the stack location of the latest
return control word. It is initially
zero.

5)   RELATIONAL.TOGGLES

This contains the current setting of
the relational toggles. It has an
initial value of zero.

2.0   FORMATS

2.1   S-Instruction Format

Each BASIC S-Instruction consists of a variable length S-Operator
followed by a variable number of arguments.

2.1.1 S-Instruction Operators

S-Operators are 3, 5 or 9 bits in length with the most frequently
used S-Operators (on a static basis) being coded with the smaller
number of bits. The length of the operator is determined by the
operator itself as follows:

```
┌──────────┐       ┌──────────┐       ┌──────────────────┐
│   000    │       │ 100   XX │       │ 111  XX   XXX    │
│   to     │       │    to    │       │                  │
│   011    │       │ 110   XX │       │                  │
└──────────┘       └──────────┘       └──────────────────┘
  3-bit operator     5-bit operator     9-bit operator
```

2.1.2 S-Instruction Arguments

Each argument can consist of a variable number of bits. The format
and interpretation of these arguments are described in detail with
the description of the individual operators.

2.2   Stack Format

The width of the Stack is 40 bits. Its length is a program parameter
determined at run time.

2.3   Data Table Format

The width of the Data Table is 40 bits. Its length is a program
parameter determined at run time.

2.4   Floating Point Binary Number Format

A floating point binary number has the following format:

```
|<-- 1 -->|<---- 9 ---->|<-------- 30 -------->|
|  SIGN   |  EXPONENT   |  NORMALIZED FRACTION  |
```

```
                       00...0 = -256
                          .
                          .
                       10...0 =   0
                          .
                          .
     0 = Negative      11...1 = +255
     1 = Positive
```
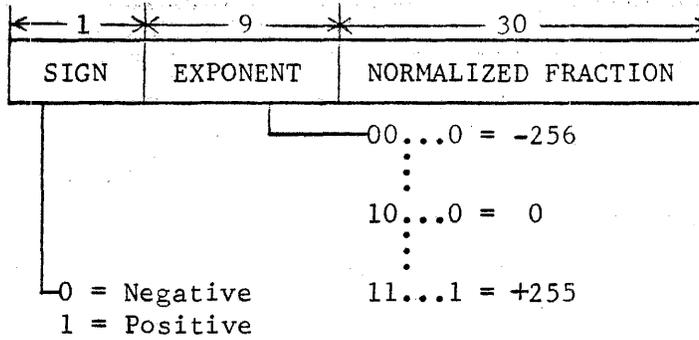
The value zero has a fractional value of zero (0...0).

2.5   Binary Integer Format

A binary integer on the stack has the following format:

```
|<-- 1 -->|<----- 9 ----->|<--------- 30 --------->|
|  SIGN   |  00000 0000    |  RIGHT JUSTIFIED INTEGER  |
```

```
  0 = Negative
  1 = Positive
```

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 6
BASIC S-LANGUAGE
P.S. #2210 0135
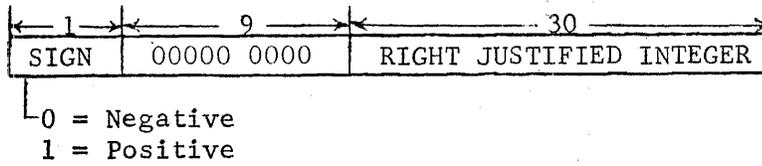
3.0   INSTRUCTION SET

Arithmetic

Name	Mnemonic	Op Code	Argument
Add	ADD	110 00	
Subtract	SUBT	110 01	
Multiply	MULT	110 10	
Divide	DIV	110 11	
Complement Sign	CSGN	111 00 110	
Set Sign Positive	PSGN	111 00 111	
Float	FLT	111 01 011	
Fix	FIX	111 01 010	
Convert to Decimal	CONV	111 10 001	
Compare Numeric	CMPN	101 11	

Stack Operations

Name	Mnemonic	Op Code	Argument
Exchange	EXCH	111 10 000	
Push	PUSH	111 10 100	NUMBER.OF.WORDS

Store Operators

Name	Mnemonic	Op Code	Argument
Store Destructive	STD	010	
Store Non-Destructive	STN	111 00 011	
Store Bits	STB	111 01 111	CHARACTER.FLAG

3.0   Cont'd.

Load Operators

Name	Mnemonic	Op Code	Argument
Load Bits	LDB	111 01 110	CHARACTER.FLAG
Large Literal	LIT	100 01	40-BIT LITERAL
Small Literal	SLIT	011	10-BIT LITERAL
Load Address	LA	000	STACK.FLAG, WORD.DISP
Load Value	LV	001	STACK.FLAG, WORD.DISP
Array Load Address	ALA	100 10	DATA.TABLE.INDEX
Array Load Value	ALV	100 11	DATA.TABLE.INDEX
Evaluate Right Subscript	ERS	111 01 000	DIMENSION.LENGTH, ROW. DIMENSION

Branch Operators

Name	Mnemonic	Op Code	Argument
For	FOR	111 01 100	NEXT, LIA, LPA, LEA
Next	NEXT	111 01 101	LIA, LPA, LEA
Case	ON	111 00 010	NO.OF.ADDRESSES, BRANCH. DISP
Bun	BUN	100 00	DISP.LENGTH, BRANCH.DISP
Branch Conditional	BXXX	101 00	BRANCH.MASK, DISP. LENGTH, BRANCH.DISP
Call	CALL	101 01	SEGMENT, DISP.FLAG, CALL. DISP
Enter	NTR	111 00 000	DISP.LENGTH, BRANCH.DISP
Return	RTN	111 00 001	STACK.CUT.VALUE
Return Value	RTNV	101 10	STACK.CUT.VALUE
Communicate	COMM	111 10 010	

Miscellaneous

Name	Mnemonic	Op Code	Argument
Load Communicate Reply	LDCR	111 10 011	
Hardware Monitor	HMON	111 10 101	

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 8
BASIC S-LANGUAGE
P.S. #2210 0135

3.1   ADD, SUBTRACT, MULTIPLY, DIVIDE

FORMAT:

OP CODE	
ADD	110 00
SUBT	110 01
MULT	110 10
DIV	111 11

Perform the indicated algebraic operation on the top two items on
the   stack and replace them on the stack with the result.

Both input items must be normalized floating point numbers.   The
result is a normalized and rounded floating point number.

In subtraction, the top most stack item is subtracted from the
second top most item.   In division, the top most stack item is
divided into the second top most item.

If the exponent of the result overflows, or underflows, or if the
divisor in division is equal to zero, a runtime error is generated.

3.2   Complement Sign (CSGN)

FORMAT:

OP CODE
111 00 110

Complement the sign (leftmost bit) of the value on the top of
the stack, i.e. change a positive value to a negative value and
vice-versa.

3.3   Set Sign Positive (PSGN)

FORMAT:

OP CODE
111 00 111

Set the sign (leftmost bit) of the value on the top of the stack
to indicate a positive value.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 9
BASIC S-LANGUAGE
P.S. #2210 0135

3.4   Float (FLT)

FORMAT:

OP CODE
111 01 011

Replace the signed 30-bit binary integer on the top of the stack with its normalized floating point number equivalent by shifting the integer Z places to the left, by supplying Z trailing zero bits and by supplying an exponent equal to $30-Z+256$. Z equals the number of leading zero bits in the integer.

3.5   Fix (FIX)

FORMAT:

OP CODE
111 01 010

Replace the floating point value on the top of the stack with a signed 30-bit binary integer, which is equal to its entire value by shifting its fractional part $30-P$ places to the right, by supplying $30-P$ leading zero bits and by changing its exponent part to zero (0...0) unless P exceeds 30 or P is negative. If P is negative, the integer zero is left on the stack. If P exceeds 30 no conversion is performed and the operation is terminated. P equals the exponent part of the floating point number minus 256.

3.6   Convert To Decimal (CONV)

FORMAT:

OP CODE
111 10 001

Replace the floating point value on the top of the stack with an unsigned five character decimal integer, which is equal to its entier value and which is coded in EBCDIC.

The results are undefined if the input value exceeds 99999_{10}.

Note: EBCDIC   0 through 9 is coded 1111 0000 through 1111 1001.

3.7   Compare Numeric (CMPN)

FORMAT:

OP CODE
101 11

Compare algebraically the top two items on the stack and set the relational toggles to indicate whether the first item is greater than (>), equal to (=) or less than (<) the second item.

Both input items must be normalized floating point numbers.

Plus and minus zero are, by definition, equal.

The two items are deleted from the stack.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 11
BASIC S-LANGUAGE
P.S. #2210 0135

3.10   Exchange (EXCH)

FORMAT:

OP CODE
111 10 000

Exchange the relative position of the top two items on the stack.

3.11   Push (PUSH)

FORMAT:

OP CODE	NUMBER.OF.WORDS
111 10 100	5 BITS

Push all occupied hardware stack registers to memory and then bump up the stack pointer by the number of words indicated by NUMBER.OF. WORDS, an unsigned 5 bit binary integer.

3.12   Store Destructive (STD)

FORMAT:

OP CODE
010

Remove the top two items from the stack and copy the second item (40 bits) removed into the location specified by the address contained in the first item removed.

The address gives the relative displacement in bits from the DATA. TABLE.BASE.

3.13   Store Non-Destructive (STN)

FORMAT:

OP CODE
111 00 011

Remove the top item containing an address from the stack and copy the next item (40 bits) from the top of the stack into the location specified by that address. The copied item remains on the stack.

The address gives the relative displacement in bits from the DATA.TABLE.BASE.

BURROUGHS CORPORATION   COMPANY CONFIDENTIAL PAGE 12
COMPUTER SYSTEMS GROUP   BASIC S-LANGUAGE
SANTA BARBARA PLANT   P.S. #2210 0135

3.14 Store Bits (STB)

FORMAT:

OP CODE 111 C1 111	CHARACTER.FLAG 0 = bits 1 = characters

Store the specified number of bits or characters from the stack into a data area.

There are four items on the top of stack, which are required by this operation and which are cut back from the stack at the completion of the operation. They are from top to bottom as follows:

1) The base relative bit address of the beginning of the data area.

2) A floating point value which is used to algebraically modify the above address. The value denotes the number of bits (CHARACTER.FLAG = 0) or the number of 8-bit characters (CHARACTER.FLAG = 1) by which the address is modified.

3) A floating point value indicating the number (0 to 40) of bits (CHARACTER.FLAG = 0) or the number (0 to 5) of 8-bit characters (CHARACTER.FLAG = 1) which are to be stored.

4) A 40-bit word containing the data, right justified, which is to be stored.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 13
BASIC S-LANGUAGE
P.S. #2210 0135

3.15   Load Bits (LDB)

FORMAT:

OP CODE 111 01 110	CHARACTER.FLAG 0 = bits 1 = characters

Load the specified number of bits or characters from a data area onto the top of the stack, right justified with leading zero fill in a 40-bit word.

There are three items on the top of stack which are required by this operation and which are replaced on the stack by the data. They are from top to bottom, as follows:

1) The base relative bit address of the beginning of a data area.

2) A floating point value which is used to algebraically modify the above address. The value denotes the number of bits (CHARACTER.FLAG = 0) or the number of 8-bit characters (CHARACTER. FLAG = 1) by which the address is modified.

3) A floating point value indicating the number (0 to 40) of bits (CHARACTER.FLAG = 0) or the number (0 to 5) of 8-bit characters (CHARACTER.FLAG = 1) which are to be loaded.

3.16   Large Literal (LIT)

FORMAT:

OP CODE 100 01	LARGE.LITERAL 40 bits

Load the 40-bit LARGE.LITERAL from the instruction onto the top of the stack.

3.17   Small Literal (SLIT)

FORMAT:

OP CODE 011	SMALL.LITERAL. 10 bits

Convert the 10-bit SMALL.LITERAL. consisting of 3 exponent bits followed by 7 fraction bits to a positive floating point value and load it onto the top of the stack.

3.18   Load Address (LA)

FORMAT:

OP CODE 000	STACK.FLAG 0 = data table 1 = stack	WORD.DISPLACEMENT S.F=0: 9 bits 1: 5 bits

Load the base relative bit address of a data table or a stack word
onto the top of the stack in the following format:

16	24
ZERO	ADDRESS
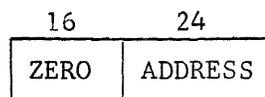
If the STACK.FLAG = 0, the address to be loaded is the address of
a data table word. In this case the WORD.DISPLACEMENT value is 9
bits long and indicates a word displacement from the base of the
data table.

If the STACK.FLAG = 1, the address to be loaded is the address of
a stack word. In this case the WORD.DISPLACEMENT value is 5 bits
long and is signed. The first bit is the sign.   The remaining
4 bits indicate a word displacement from the current setting of
the return control register. If the sign bit = 1, the word dis-
placement is a positive word displacement. If the sign bit = 0,
the word displacement is a negative word displacement.

3.19   Load Value (LV)

FORMAT:

OP CODE 001	STACK.FLAG 0 = data table 1 = stack	WORD.DISPLACEMENT S.F=0: 9 bits 1: 5 bits

Load a 40-bit value from the data table or the stack to the top of
the stack.

This operator is functionally the same as the load address operator
(see section 3.18) except the actual word, rather than the address
of that word, is loaded onto the stack.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 15
BASIC S-LANGUAGE
P.S. #2210 0135

3.20   Array Load Address (ALA)

FORMAT:

OP CODE	DATA.TABLE.INDEX
100 10	9 bits

Load the base relative bit address of an array element to the top of the stack and insure that the array is present in memory.

The DATA.TABLE.INDEX, an unsigned binary integer, is multiplied by 40 and used as a displacement into the data table to locate an array descriptor with the following format:

1	1	2	18	12	6
DD.BIT	STRING.BIT	NOT USED	BOUNDS	SEGMENT	DISPLACEMENT

The floating point number on the top of the stack is an array subscript which is truncated to an integer and compared to the bounds field of the array descriptor and with zero. If this converted subscript is less than zero or greater than or equal to the bounds value, a runtime error is generated; otherwise, it is added to the segment and displacement fields (taken as one 18 bit field) of the array descriptor. The resultant 12 bit segment value is used to index into the program's data dictionary, and the resultant 6 bit displacement value, multiplied by 40, is added to the memory address taken from that data dictionary entry to produce the base relative address of the desired array element. This address replaces the original subscript value on the top of the stack.

The format of a data dictionary entry is:

1	1	1	2	28	31
SY.LOCK	SY.MEDIA	SY.IN.PROCESS	UNUSED	SY.ADDRESS	

    ┌─ 0   SY.ADDRESS = DISK ADDRESS
    └─ 1   SY.ADDRESS = MEMORY ADDRESS

3.21   Array Load Value (ALV)

FORMAT:

OP CODE	DATA.TABLE.INDEX
100 11	9 bits

Load a 40-bit array element to the top of the stack.

This operator is functionally the same as the array load address operator (see section 3.20), except that the actual array element, rather than the address of that element, replaces the subscript on the top of the stack.

3.22   Evaluate Right Subscript (ERS)

FORMAT:

OP CODE	DIMENSION.LENGTH	ROW.DIMENSION
111 01 000	2 bits	4,7,10 or 18 bits

Convert the two subscripts for a two dimensional array into a single
subscript suitable for use by the array load operators.

The two values on the top of the stack are as follows:

> Top word: Column subscript
> 2nd word: Row subscript

DIMENSION.LENGTH specifies the container size of ROW. DIMENSION
as follows:

> 00 = 4 bits
> 01 = 7 bits
> 10 = 10 bits
> 11 = 18 bits

ROW.DIMENSION is a binary integer which specifies the number of
40-bit words in an array row.

The two stack subscript values are first truncated to an integer.
Then the product of the row subscript times the ROW.DIMENSION is
added to the column subscript. This result is then converted to a
normalized, rounded floating point subscript value which replaces
the two original subscript values on the stack.

In addition to any errors which may be encountered in the arithmetic
routines (add and multiply), a runtime error will be generated if
the column subscript is found to be less than zero or greater than
or equal to the ROW.DIMENSION value.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 17
BASIC S-LANGUAGE
P.S. #2210 0135

3.23   For (FOR)

FORMAT:

OP CODE 111 01 100	OP CODE (NEXT) 111 01 10	LOOP INDEX ADDRESS 9 bits	LOOP PARAMETERS ADDRESS 9 bits	LOOP ENDING ADDRESS 18 bits

Store the step size and then the ending index value into the data
table at the location specified by the loop parameters address and
then compare the beginning index value with the ending index value.

The following three floating point values are required to be on the
top of the stack. They are removed at the end of the operation:

        Top Word: Step Size
        2nd Word: Ending Index Value
        3rd Word: Beginning Index Value

The loop is satisfied if, for a positive step size, the beginning
index value is greater than the ending index value; or, for a nega-
tive step size, the beginning index value is less than the ending
index value.

If the loop is not satisfied, the beginning index value is stored
in the data table at the location specified by the loop index address
and the instruction following the FOR instruction is then selected.
If the loop is satisfied, the next instruction is selected from the
code location specified by the loop ending address.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 18
BASIC S-LANGUAGE
P.S. #2210 0135

3.24   Next (NEXT)

FORMAT:

OP CODE 111 01 101	LOOP INDEX ADDRESS 9 bits	LOOP PARAMETERS ADDRESS 9 bits	LOOP ENDING ADDRESS 18 bits

Add the current loop index value and the step size to form a new loop index value and then compare this new loop index value with the ending index value.

The current loop index value is found in the data table at the location specified by the loop index address. The step size and the ending index value, in that order, are found in the data table at the location specified by the loop parameters address.

All values are in floating point format.

The loop is satisfied if, for a positive step size, the new loop index value is greater than the ending index value; or, for a negative step size, the new loop index value is less than the ending index value.

If the loop is not satisfied, the new loop index value is stored in the data table at the location specified by the loop index address and the instruction following the FOR instruction is then selected. If the loop is satisfied, the next instruction is selected from the code location specified by the loop ending address.

3.25   ON (ON)

FORMAT:

OP CODE 111 00 010	NUMBER.OF. ADDRESSES 6 bits	BRANCH.DISPLACE- MENT.1 18 bits	...	BRANCH.DISPLACE- MENT.N 18 bits

Select the next instruction from the location determined by adding the unsigned binary integer BRANCH.DISPLACEMENT to the base address of the current code segment.

The particular BRANCH.DISPLACEMENT used is determined by the integer value of the floating point number found on the top of the stack. This item is deleted from the stack.

If this floating point value after conversion to a signed integer (see section 3.5 Fix), is less than one or greater than the unsigned 6-bit binary integer given by NUMBER.OF.ADDRESSES, a runtime error is generated.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 19
BASIC S-LANGUAGE
P.S. #2210 0135

3.26   Branch Unconditionally (BUN)

FORMAT:

OP CODE 100 00	DISPLACEMENT.LENGTH 1 bit	BRANCH.DISPLACEMENT 12 or 18 bits

Select the next instruction from the location determined by adding the unsigned binary integer BRANCH.DISPLACEMENT to the base address of the current code segment.

If DISPLACEMENT.LENGTH = 1, the length of the BRANCH.DISPLACEMENT value is 18 bits; otherwise, it is 12 bits long.

3.27   Branch Conditional (Bxxx)

FORMAT:

OP CODE 101 00	BRANCH.MASK 3 bits	DISPLACEMENT.LENGTH 1 bit	BRANCH.DISPLACEMENT 12 or 18 bits

Compare the BRANCH.MASK to the current setting of the relational toggles. If any "one" bit in the BRANCH.MASK is matched by a corresponding "one" bit in the relational toggles, select the next instruction from the location determined by adding the unsigned binary integer BRANCH.DISPLACEMENT to the base address of the current code segment; otherwise terminate the instruction.

The following table gives the binary values of the BRANCH.MASK required to implement the given branch type:

BNUN = 000 = NO OPERATION (NO BRANCH),

BGTR = 001 = GREATER THAN,

BLSS = 010 = LESS THAN,

BNEQ = 011 = NOT EQUAL,

BEQL = 100 = EQUAL,

BGEQ = 101 = GREATER THAN OR EQUAL,

BLEQ = 110 = LESS THAN OR EQUAL, and

BANY = 111 = BRANCH UNCONDITIONALLY UNLESS THE RELATIONAL
              TOGGLES ARE UNINITIALIZED (= ZERO).

If DISPLACEMENT.LENGTH = 1, then the length of the BRANCH.DISPLACEMENT value is 18 bits; otherwise, it is 12 bits long.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 20
BASIC S-LANGUAGE
P.S. #2210 0135

3.28   Call (CALL)

FORMAT:	OP CODE	CALLED.SEGMENT	DISPLACEMENT.FLAG	CALL.DISPLACEMENT
	101 01	7 bits	1 bit	0 or 16 bits

Branch to a subroutine in the specified code segment after forcing all
occupied hardware stack registers to memory and after placing a return
control word onto the top of the stack in memory.

The format of the return control word is as follows:

15	7	18
RETURN CONTROL REGISTER	SEGMENT	DISPLACEMENT

The current contents of the return control register, and the segment
and displacement pointing to the location of the next in line s-operator
following the call is placed into the return control word. This return
control word will be used by a return or return value operator to exit
from the subroutine back to the next in line s-operator following the
call.

The return control register is, after saving its contents in the return
control word, updated to point to the location in the stack of the newly
formed return control word.

If DISPLACEMENT.FLAG = 1, then the 16-bit CALL.DISPLACEMENT field
is present; otherwise, a DISPLACEMENT value of 48 is assumed.

The actual branch is performed by using the CALLED.SEGMENT field
as an index into the code segment dictionary and then adding the
absolute address found therein to the CALL.DISPLACEMENT to obtain
the address of the next operator to be executed.

3.29   Enter (NTR)

FORMAT:	OP CODE	DISPLACEMENT.LENGTH	BRANCH.DISPLACEMENT
	111 00 000	1 bit	12 or 18 bits

Branch to a subroutine or a function routine in the current code segment
after forcing all occupied hardware stack registers to memory and after
placing a return control word onto the top of the stack in memory.

If DISPLACEMENT.LENGTH = 1, then the length of the BRANCH.DISPLACEMENT
field is 18 bits; otherwise, it is 12 bits long.

See the "Call" operator (section 3.28) for a description of the enter
operation. The operations are identical. Only the format of the
instruction differs.

3.30   Return (RTN)

FORMAT:

OP CODE	STACK.CUT.VALUE
111 00 001	0000 to 1111

Return from a subroutine or a function routine by using the return con-
trol register to obtain the latest return control word and to cut the
stack back to just below it. Then cut the stack back beyond this point
by the number of words indicated by STACK.CUT.VALUE.

From the return control word restore to the return control register
the pointer to the location of the previous return control word and
use the code segment and displacement fields to locate the next in-
struction to be executed.

3.31   Return Value (RTNV)

FORMAT:

OP CODE	STACK.CUT.VALUE
101 10	0000 to 1111

Save the item on the stack found immediately on top of the latest
return control word, perform a return (see section 3.30), and then
place the saved item on the top of the stack.

BURROUGHS CORPORATION
COMPUTER SYSTEMS GROUP
SANTA BARBARA PLANT

COMPANY CONFIDENTIAL   PAGE 22 - FINAL
BASIC S-LANGUAGE
P.S. #2210 0135

3.32 Communicate (COMM)

FORMAT:

OP CODE
111 10 010

Load the 48 bit RS.COMMUNICATE.MSG.PTR area of the RS.NUCLEUS with an SDL type data descriptor indicating a length of 120 bits and containing the absolute address of the data table. The first 120 bits of the data-table is reserved for use as a communicate message area.

The format of this SDL type data descriptor is as follows:

8	16	24
Type	Length	Absolute Address
0100 0000	0...01111000	of data-table

Store the status of the M-Machine in the appropriate parts of this program's RS.NUCLEUS.

Instate the program (MCP) whose RS.NUCLEUS address is given in the RS.COMMUNICATE.LR field of this program's RS.NUCLEUS.

3.33 Load Communicate Reply (LDCR)

FORMAT:

OP CODE
111 10 011

Load the low order 40 bits of the 48 bit RS.REPLY area of the RS.NUCLEUS to the top of the stack.

3.34 Hardware Monitor (HMON)

FORMAT:

OP CODE
111 10 101

Remove the floating point number on the top of the stack and convert it to an integer (see FIX Section 3.5). Use the lower order 8 bits of the resultant integer as variants in the hardware monitor micro instruction.