RICE UNIVERSITY COMPUTER

SPIREL SYSTEM

ASSEMBLY SYSTEM

APRIL, 1964

The

S P I R E L S Y S T E M

for the

R I C E U N I V E R S I T Y C O M P U T E R

Programming development on the Rice University Computer has been supported by The National Science Foundation under grants G-7648 and G-17934. Construction of the computer was supported by The United States Atomic Energy Commission under contracts AT-(40-1)-1825 and AT-(40-1)-2572.

April, 1964

SPIREL SYSTEM

SPIREL SYSTEM

I. The Concepts

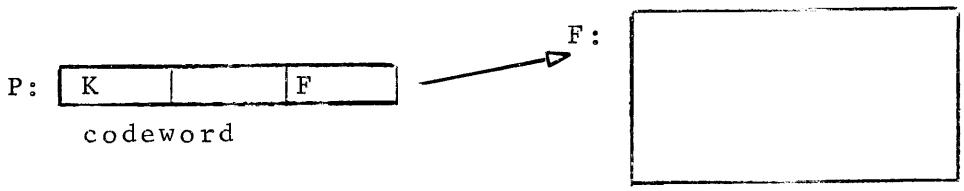
A computer serves a user by decoding instructions and performing the operations specified. In an analagous manner, the SPIREL system serves a user by decoding control words and performing the operations specified. In fact, once in the machine SPIREL may be thought of as an extension of the Rice Computer.

As instructions are used to dictate computer operations on single words in the memory, so are control words used to dictate SPIREL operations on blocks. Physically, a block is a set of contiguous words in memory. Associated with each block is a one-word "label" called a codeword. A block is a logical unit from the point of view of the user; it may contain a program, a vector of data, or codewords which in turn label other blocks. In general, an array is a logical structure which consists of a codeword which labels a block of codewords which label blocks, and so on until on the lowest level are blocks which do not contain labels. The depth or dimension of an array is just the number of codeword levels in the array. A program is a single block with one codeword, a one-dimensional array. A data vector is a single block with one codeword, a one-dimensional array. A matrix of data is a vector of data vectors, a two-dimensional array. Collections of programs and data vectors may be logically grouped to form program and data arrays of any depth deemed organizationally useful to the SPIREL user.

An array is uniquely associated with its single highest codeword, the primary codeword. In most applications all addressing of information in arrays (contents in lowest level blocks) is done through the primary codeword. Thus, access to information in an array depends on only one address, the codeword address for the array. The physical location of blocks is irrelevant to the user, so allocation of storage for blocks is performed by SPIREL, and addressing through levels of codewords constructed by SPIREL is accomplished by the indirect addressing of the hardware. A fixed

region of the memory, locations 200 through 277 (octal), is by convention reserved for primary codewords, and allocation in this area is the responsibility of the individual user. The unique correspondence of a primary codeword address to its array provides an informative "name" for the array. A program with codeword address 225 may be called program *225, and a matrix with primary codeword at 271 may be called matrix *271; the '*' symbolizes indirect addressing in the assembly language, and here serves to emphasize this operation in connection with codewords.

A program *P of length K may occupy a block beginning at machine address F. Then program *P is represented in the machine as



In programming, control is passed to this program by the code

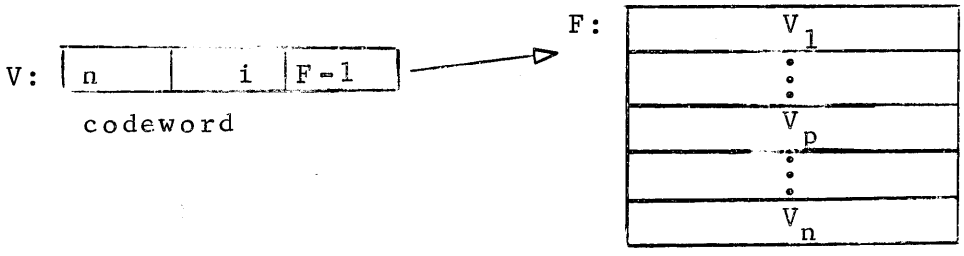
```
TSR     *P
```

which becomes

```
TSR     F
```

when the hardware indirect addressing is carried out. The address formed is that of the first word of the program.

A vector *V of n data elements V_1, V_2, ..., V_n may occupy a block beginning at machine address F. Then vector *V is represented in the machine as



In programming, the data element V_p is addressed by the code

```
p  →  index register i
```

then

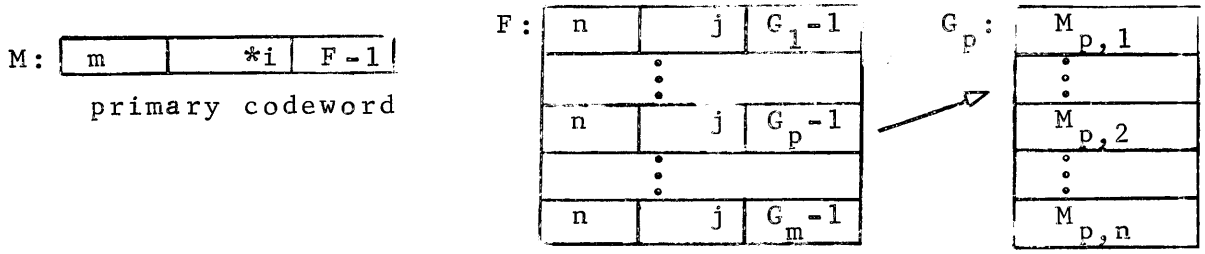
```
operation*V
```

which becomes

operation p+F-1

when the hardware indirect addressing is carried out. The address formed is that of V_p, as desired. The standard vector form uses index register B1 for element addressing. Non-standard forms permit variability of indexing and even allow the first word of the block containing the vector elements to be addressed as V_K where K is any integer.

A matrix *M of m rows by n columns of data $M_{1,1},\ldots,M_{m,n}$ is stored one row per block and is represented in the machine as



In programming the data element $M_{p,q}$ is addressed by the code

p → index register i

Q → index register j

then

operation *M

which becomes

operation *p+F-1

and then

operation q+G_p-1

when two levels of hardware indirect addressing are carried out. The address formed is that of $M_{p,q}$, as desired. This addressing in no way depends on the size of the matrix *M. The standard matrix form is for a rectangular matrix, using B1 for row specification and B2 for column specification. Non-standard forms permit variability of indexing and non-rectangular structures and even allow the first element to be addressed as $M_{p,q}$ for p and q any integers.

SPIREL operations on arrays are dictated by control words

which specify the primary codeword address for the array. Such operations are:

- to take space for a program, vector, or standard matrix
- to print the lowest level blocks in an array
- to punch an array
- to execute a program
- to free the space occupied by an array
- and many others

Of particular importance is the fact that arrays may be dynamically created and erased or changed in size and structure so that only immediately pertinent arrays occupy space at any time during the run of a user's system.

SPIREL control words are 18-place octal configurations, i.e., one machine word in length. With the SPIREL system in the machine, control words may be transmitted to the system in two ways:

- internally under program control by the user --

  control word → T7

  and TSR *126

- externally to the SPIREL communication routine from paper tape --

  control word preceded by 'carriage return'
  punch on paper tape in the reader

  or from the typewriter --

  control word $\overset{\text{type}}{\rightarrow}$ U-register.

In all these cases the control word is in fact transmitted to XCWD (Execute Control Word), program *126 in the SPIREL system. This program is the nucleus of SPIREL; it interprets each control word and may use other programs in the system to carry out specified operations. SPIREL is, then, a collection of programs, and any of these may be utilized directly by the user of the system.

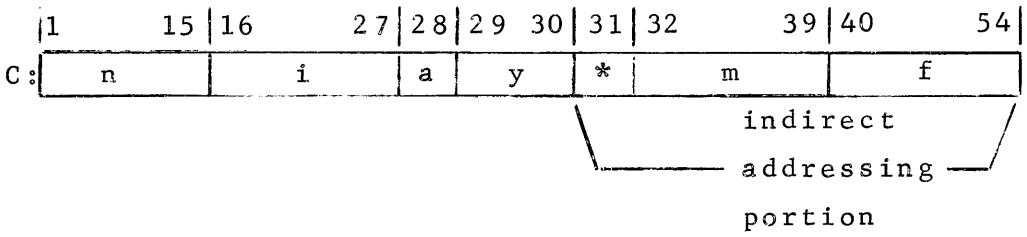
Details about codewords, system organization, control word decoding and formats, storage control, and the SPIREL programs are given in the succeeding sections.

II. Codewords

With every block of memory is uniquely associated a codeword which has two primary functions:

- description of the block, including current length and location, current type of block content, and printing format for the block
- indirect addressing portion appropriate for programmed addressing through the codeword into the block.

The format for a codeword at address C, which labels a block beginning at address F=f+i is:



where

n=length of the block.

i=empty (+0) if no B-mods in codeword, in particular if block contains a program;
initial index in 1's complement format if B-mods in codeword, in particular if block contains a vector; first word of block is addressed as C_i; standard SPIREL provision is for i=1; i=0 is represented by i=7777.

a=1 if block contains codewords; empty (0) otherwise.

y=printing format to be used for output of block if none given in print control word
0: octal, 4 words per line (standard SPIREL provision)
1: hexad, 108 characters per line
2: octal, 1 word per line in program layout
3: decimal, 5 words per line

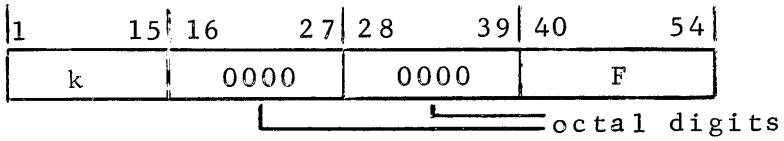
*,m=indirect addressing and B-modification bits, effective in addressing indirectly through the codeword.

f=F-i, F=the address of the first word of the block.

A codeword is completely formed by SPIREL at the time the corresponding block is created. This creation is the result of a control word to take space (structure formed and lowest level

blocks filled with zeroes) or one to read a block (structure formed and lowest level blocks filled with words read from paper tape). The most frequently used codeword forms are illustrated below.

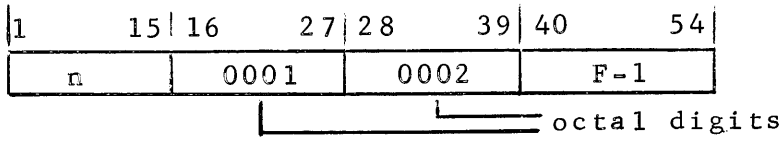
- For a program *P of length k:



where the first word of the program is located at address F. Control is passed to the program by the code

```
TSR     *P
```

- For a standard vector *V of length n:

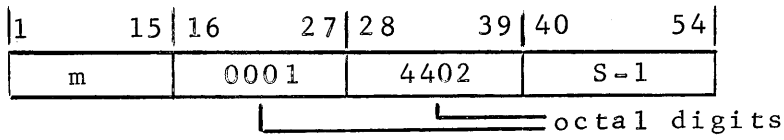


where the vector elements V_1 is located at address F, the initial index=1, and a B1 modifier is used. The element V_p is addressed by the code

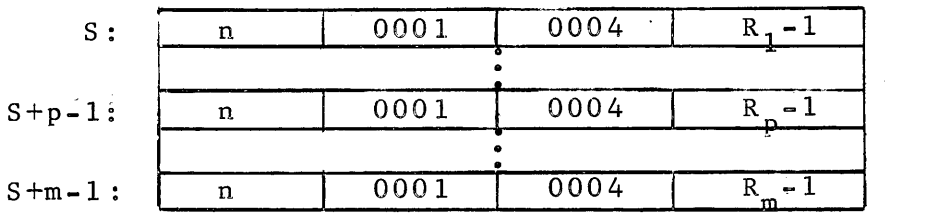
```
SB1     p
CLA     *V
```

- For a standard matrix *M of m rows by n columns:

primary codeword



secondary codewords



where each row is stored in a separate block and the matrix element $M_{p,1}$ is located at address R_p-1 for each p, the initial row and column indices=1, a B1 modifier is used for row specification, a B2 modifier is used for column specification, the primary codeword contains an a-bit and a *-bit. The element $M_{p,q}$ is addressed by

the code

```
SB1     p
SB2     q
CLA     *M
```

For any array the primary codeword address is specified by the user, but the location of blocks in the memory is left to the "discretion" of SPIREL. Any given block may be located variously from run to run, and may be moved even during a run if the STEX storage control mechanism in SPIREL is active and such manipulation is necessary for another requested allocation.

III. SPIREL System Organization

III.1. Memory Utilization

The SPIREL system itself is a collection of programs, tables, and individual constants. System conventions provide memory utilization as follows:

octal addresses	use
00000-00007	machine full-length fast registers Z,U,R,S,T4, T5,T6,T7
00010	not used
00011-00017	machine trap locations
00020-00026	SPIREL external communication routine (explained below)
00027-00077	used only by SPIREL program *120, diagnostic dump
00100-00177	system codewords and individual constants
00200-00277	region for primary codewords of the system user
00300-17577	storage of blocks labelled by system or user codewords, each block containing a program, data or codewords which in turn label other blocks
17600-17777	working push-down storage addressed by index register B6 (explained below)
20000-20007	entry to SPIREL program *120, diagnostic dump.

III.2. External Communication

The SPIREL external communication routine accepts directives from machine location 23 (octal). This location contains a 'halt and transfer' instruction. At this stop, the user at the console may type a SPIREL control word into U or ready a paper tape bearing a control word preceded by a 'carriage return' punch in the reader. Pushing CONTINUE causes the available control word to be communicated to SPIREL, which will then decode it and carry out the specified action. If a single paper tape contains a series of control words, FETCHing at the external communication stop will cause control words to be read from paper tape and executed with no stops between actions

specified until the end of the tape or a null control word is encountered.

III.3. B6-list

The working push-down storage area is addressed by index register B6 and is commonly called the B6-list. SPIREL programs use the B6-list; programs of the system user may similarly use the B6-list; and index register B6 may be used for other purposes only if the working storage setting is maintained for those programs which depend on it.

Conventional use of the B6-list depends on one fact: that B6 contains the address of the first word of a block of storage not in use. Therefore, if one word of temporary storage is required, it is taken at B6 and B6 is incremented by one; if the last word stored on the B6-list is retrieved from the address B6-1 and is in fact no longer resident on the list, B6 is decremented by 1, and the storage location may be reused. The initial system setting of B6 is to 17600 (octal), and the push-down storage area extends to address 17777 (octal).

A frequent application of the B6-list is for temporary storage of fast registers to be used by a subroutine, but to appear undisturbed to the program using the subroutine. A program wishing use but preserve T4, T5, and T6 might use the B6-list as follows:

upon entry

```
        T4      STO     B6,B6+1
        T5      STO     B6,B6+1
        T6      STO     B6,B6+1
```

computation with private use of T4, T5, T6 and any desired use of the B6-list

prior to exit

```
        LT6     B6-1,B6-1
        LT5     B6-1,B6-1
        LT4     B6-1,B6-1
```

exit with B6 setting same as that upon entry.

III.4. System Components

The programs, tables, and individual constants which comprise the SPIREL system are listed below. The programs are fully explained in later sections, and the diagram of SPIREL component linkage shows how the various components are functionally interconnected.

INDIVIDUAL CONSTANTS

Address	Name	Function
100	STORAG	describes available storage
114	PRCT	current active length of ADDR
115	ACWD	used by PUNCH
117	STPNT	gives current active lengths of ST and VT
121	FWA	first word address of last program tagged (used by TRACE and TAGSET)
124	NAME	symbolic name (if any) of block currently being operated on by SPIREL

VECTORS and PRINT MATRIX

Codeword Address	Name	Use	Length$_8$
113	ST	Symbol Table	100
116	PM	Print Matrix	200
122	VT	Value Table	100
125	ADDR	Base Address Vector	6

PROGRAMS

Codeword Address	Name	Use	Length$_8$
13	TRACE	Repeat Mode Trace	251
110	HDPR	Print Control Word	55
111	MATRX	Process Matrix	135
120	DIADMP	Diagnostic Dump	57
126	XCWD	Execute from Control Word	276
127	SETPM	Set Up Print Matrix	75
130	SMNAM	Find Symbolic Name	26
131	DATIME	Print Date and Time	14
132	CLOCK	Decode Clock	55
133	PCNTRL	Punch Control Word	15
134	FIELD	Set Up Word in Program Format in Print Matrix	20
135	TAKE	Take Memory Space	5
136	SAVE	Save Fast Registers	41
137	UNSAVE	Unsave Fast Registers	41
140	DELETE	Insert or Delete Space	124
141	CHINDX	Change Initial Index	13
142	TAGSET	Tagset	15
143	CHN	Take Chain Storage	13
144	PRINT	Print	70
145	PUNCH	Punch	130
146	XCWSQ	Execute Control Word Sequence	15
147	SYMRF	Load Symbolic Cross Reference	12
150	PRNWTG	Set Up Tag in Print Matrix	16
151	PRSYM	Print Symbol and Value Tables	43
152	PWRTN	Conversion of Powers of Ten	43
153	MRDDC	Multiple Read Decimal	144
154	SETX	Storage Exchange	213
155	BINDC	Binary to Decimal Conversion	102
156	RDCHK	Read with Check-Sum	35
157	PUNCHK	Punch with Check-Sum	24
176	TLU	Table Look-Up	23

IV. Control Words

IV.1. Program *126, XCWD

The nucleus of the SPIREL system is the program *126, XCWD (Execute from Control Word). This routine interprets control words received in T7 and carries out the specified operations. The work of *126 may in most cases be described in the following steps:

1) address determination -- consists of determining the address of the first word to be operated upon and the number of words to operated upon
2) operation determination -- consists of determining the operation to be performed
3) operation execution -- consists of performing the operation or using another SPIREL program to carry out the operation

XCWD is 276 (octal) words long. It accepts a control word in T7 and disturbs no other fast registers. Two sense lights affect the behavior of *126:

SL14 off causes XCWD to print one line for each control word it executes. The information printed includes the control word, the operation, the name of the block operated on, the location and number of words operated on, and the number of free words of storage remaining. This SPIREL monitoring provides useful load records and may help in debugging. A system which has been checked out would probably run with SL14 on to suppress monitoring.

SL15 on causes "reading" and "correcting" operations of SPIREL to bypass storage of what is read and instead compare it with what is currently in the locations where storage would otherwise take place.

IV.2. Control Word Format

A control word is divided into seven fields: N,w,x,y,z, R, and F. These are arranged as follows:

bits

1 15	16 27	28 39	40 54
nnnnn	wxyz	rrrr	fffff
N		R	F

where each lower case letter represents one octal digit. In general, N,x,R, and F concern address specification; w,y, and z concern operation specification.

IV.3. Address Specification

General rules for address specification may be stated; exceptions exist and are noted in the list of control words later in this section.

The control word fields N,x,R, and F provide information which determines

$\mathcal{F}$, the address of the first word to be operated on

and η, the number of words to be operated on.

To specify SPIREL operation on a set of locations whose absolute machine locations are known:

x=0
F=$\mathcal{F}$
N=η
R irrelevant

For example, it might be useful to have SPIREL print the user's codeword region, 100 (octal) words starting at location 200 (octal).

To specify SPIREL operation on all of or a portion of a block labelled by a codeword:

x=1

If the codeword address is known:

F=codeword address

If the block is named:

F=0

and the name is given after the control word on paper tape or in T4 for internal programmed communication (see Genie Notes, section on Symbolic Addressing in SPIREL). If $\mathcal{F}$ is to be the first word of the block or blocks operated on:

R=0

otherwise, R specifies the relative program word (counting from 1) or the relative vector element (counting from the initial index of the vector, 1 for standard vectors) whose address is to be taken as $\mathcal{F}$. Note that the 0^{th} element of a vector is specified by R=7777. If $\mathcal{N}$ is to be the current length of the block or blocks operated on:

N=0

otherwise:

N=$\mathcal{N}$

To specify SPIREL operation on the entire contents of each of the lowest level blocks of an array which is labelled by a codeword:

x=4
N=0
R=0

(More general application of x=4 is explained in the section on Recursive Application of SPIREL.)

IV.4. Basic SPIREL Operations

The most basic SPIREL operations and the corresponding control word forms are described in this section. This set is sufficient for initial understanding and use of the system.

Read

Read operations are specified by control words with w=0 and x=0,1,2, or 4. The y field specifies the read mode:

- y=0, octal from paper tape -- 18 octal digits per word, each preceded by a carriage return
- =1, hexad from paper tape -- 9 hexads per word, each preceded by a carriage return
- =2, zeros generated by SPIREL
- =3, decimal from paper tape, each word followed by a carriage return and in the form discussed in the section on paper tape input formats under Use of SPIREL
- =4, hexad with tags and checksum from paper tape in the form punched by SPIREL

For y=0,1 or 2 words are stored with the tag given by z. As words

are read (except SPIREL-generated zeroes, specified by y=2), they are checked for proper storage in memory. If the check fails, the word as stored and the location of the word are typed in octal on the console typewriter. The location given is an absolute machine address if x=0 in the control word; it is the relative location in the block if x≠0 in the control word. If SL15 is on, the words read (except SPIREL-generated zeroes, specified by y=2) are not stored but are compared to the contents of the memory location where storage would normally take place. If the comparison fails the word actually stored in memory and its location are typed on the console typewriter as above.

Read control word forms are as follows:

nnnnn 00yz 0000 fffff Read N words in mode y and store with tag z (if y=0,1, or 2) beginning at location F.

nnnnn 01yz rrrr fffff If codeword at F addresses an array and STEX storage control is active, free that array. Create a block of length N and form its codeword at F. Place octal configuration given by R in the corresponding bit positions of codeword at F. (The last three triads of R specify the B-modification and indirect-addressing bits to be used. The first triad describes the contents of the block being read; its interpretation is described under Codewords.) If no B-modification is specified, set initial index=0; otherwise set initial index=1. Read N words in mode y and store with tag z (if y=0,1, or 2) into the block labelled by the codeword at F.

nnnnn 02yz 0000 fffff Exactly equivalent to the control word nnnnn 01yz 0002 fffff, used to read a standard vector of data.

nnnnn 04yz rrrr fffff If codeword at F addresses an array and STEX storage control is active, free that array. Create the structure of a standard matrix with N rows and R columns with primary codeword at F. Read N×R words, successive complete rows, in mode y with tag z (if y=0,1, or 2) into the matrix with primary codeword at F.

Correct

The correct operation is specified by a control word with w=1 and x=1,2, or 4. Correction implies that some part of an existing unit is to be replaced with new information without completely recreating the unit. Therefore, a correct is just a read into an existing block over the previous contents. The fields y and z specify mode and tag as explained for read operations, and again SL15 on causes comparison instead of storage of what is read.

The correct control word form is as follows:

nnnnn 1lyz rrrr fffff Read N words into the block labelled by the codeword at F, where N and R are specified according to the standard rules for address specification. Read in mode y with tag z (if y=0,1, or 2).

Tagset

Tagset operations are specified by control words with w=2. The purpose is to set tags on words in the memory, and the tag to be set is given by z=0,1,2, or 3 (tag 0 meaning no tag). If y=3, all words in the address range specified are tagged. Otherwise, all words in the address range specified are considered instructions and tagging is selective, with a word being tagged only if its "class" triad = y. (The class of an instruction is given in the third triad from the left.)

The tagset operation is most often used to set tag 3 on instructions in programs to be executed. Then if execution is carried out in the trapping mode, the trace program in SPIREL monitors on the printer the execution of the tagged instruction. This trace output is explained in detail in discussion of the SPIREL trace program *13.

The tagset control word forms are as follows:

nnnnn 20yz 0000 fffff Set tag z on all words of class y from location F to location F+N-1, inclusive.

nnnnn 21yz rrrr fffff Set tag z on all words of class y in the block labelled by the codeword at F, where N and R are specified according to the standard rules for address specification.

Execute

The execute operation is specified by wxyz=3100 and is designed so that SPIREL will, in essence, transfer control to the address specified as the entry to a closed subroutine. This operation is usually employed as an external directive to SPIREL. Primary control is then with SPIREL; successive programs may be executed, with other SPIREL operations interspersed as desired.

The form of the execute control word is as follows:

00000 3100 rrrr fffff Transfer control to the R^{th} word of the program which comprises the block labelled by the codeword at F. At entry to the specified program, all fast registers except PF are set to the values they had at the time the control word was given to SPIREL. The program executed should be written as a closed subroutine, i.e., it should exit to the address contained in PF upon entry.

Print

Print operations are specified by control words with w=4. The format of output is given by y as follows:

y	format	words/line
0	octal	4
1	hexad	12 (108 chars.)
2	octal, program format	1
3	decimal	5
4	octal, for $8\frac{1}{2}$x11" pages	3
5	octal, with tag	1
6	decimal, with tag	1
7	decimal, for $8\frac{1}{2}$x11" pages	3

For all options except y=1 (hexad), the octal location of the first word on each line is printed at the left of the line. This number is the relative location in a block for x=1 or 4 (relative) or the absolute machine address for x=0 (absolute).

The format for each decimal number printed is:

floating point

±d.ddddddddddde±dd

12 decimal digits; exponent, 2 decimal digits

fixed point integer

±ddddddd

1-15 decimal digits

The SPIREL monitoring provided if SL14 is off provides a printed identifier with each block printed.

The print control word forms are as follows:

nnnnn 40y0 0000 fffff Print in format y N words beginning at location F.

nnnnn 41y0 rrrr fffff Print in format y from the block labelled by codeword at F, where N and R are specified according to the standard rules for address specification.

00000 44y0 0000 fffff Print in format y all of the lowest level blocks in the array labelled by codeword at F.

Punch

Punch operations are specified by control words with w=5. The punch format is given by y and corresponds to the mode in which the information punched will later be read:

y=0, octal
=1, hexad
=2, no information (zeros to be generated internally at read time to correspond to information specified in punch control word)
=4, hexad with tags and checksum

If SL14 is off, SPIREL monitoring is provided and every item punched is preceded and followed by control words sufficient to cause the information punched to be read at a later time into logical position identical to that at time of punching; this is the usual procedure. If SL14 is on, no control words accompany the information punched.

If SL14 is off and a complete array is punched, the control words punched cause recreation of an identical array when the punched tape is later read. If SL14 is off and only part of an array is punched, the control words punched assume that an identical structure exists when the punched tape is later read, and the punched information is corrected into the structure.

Any tape which is punched with the SPIREL punch operations may then be read under SPIREL control with SL15 on to effect a validation of the punched tape by comparison with the information that was to be punched.

The punch control word forms are as follows:

nnnnn 50y0 0000 fffff Punch in mode y N words beginning at location F.

nnnnn 51y0 rrrr fffff Punch in mode y from the block labelled by the codeword at F, where N and R are specified according to the standard rules for address specification.

00000 54y0 0000 fffff Punch in mode y all of the array labelled by the codeword at F.

Activate STEX

The simple storage control algorithm in SPIREL operates on a principle of linear consumption of space in memory. If STEX, the storage exchange program, is active at the time blocks are created, later redefinition of these blocks will result in the space previously occupied being returned to the system for re-use. Thus, at any given time only space which is currently labelled by a codeword is in use. Activation of STEX causes the STEX domain to be defined from the extent of linear consumption through location 17577, or up to the B6-list region. Any block created after STEX is activated is said to be loaded under STEX control and will be located in the STEX domain. Any block which is to be recreated under STEX control must be created initially under STEX control. The STEX storage control system is described in detail in another section.

The control word which causes STEX to be activated is

00000 3120 0000 00135

It belongs to the execute class of control words. Once STEX is activated, subsequent activations are meaningless but harmless.

Print, Punch from Console

For convenience in external communication to SPIREL by typing into U at the console, an abbreviated control word of the form

00000 0000 00000 fffff

is defined. Here F is the codeword address of an array. If SL15 is off, all of the lowest level blocks in the array labelled by the codeword at F are printed in the format specified by bits 29 and 30 in F:

binary 00, octal
01, hexad
10, octal -- program format
11, decimal

If SL15 is on, all of the array labelled by the codeword at F is punched in hexad with tags and checksum.

IV.5. More SPIREL Operations

The SPIREL operations described in this section are not necessary for initial understanding and use of the system.

Chain Storage Generation

The control word

nnnnn 0300 0000 fffff

causes a chain of N words to be formed from address F. Beginning at F, the first 15 bits of each word contains the address of the next word in the chain, the last word in the chain being "linked" to F. The remainder of each word is cleared and F is given a tag 1. This control word must not be used with STEX active as the block containing the chain must not be located in the STEX domain.

Address Base Manipulation

SPIREL operations may be applied to blocks and arrays when the pertinent codeword address is known. The primary codeword address of any array is fixed and known to the user, and is in

the range 200-277 (octal). When the F fields of control words with x=1 are interpreted as machine addresses, the SPIREL address base is said to be set to zero. For the purpose of applying SPIREL operations to sub-arrays, the SPIREL address base may be set so that the F field of control words is interpreted relative to the first word of a block of codewords whose codeword contains no B-modifications or to the 0^{th} element of a block of codewords whose codeword contains B-modifications. Each base change in a sequence of base changes will progress one level into an array, and such a sequence may go to depth 5.

Consider an array of three dimensions with primary codeword at address A and B-modification on each level. Assuming that A exists in the machine, the following series of SPIREL operations illustrates the use of address base manipulations:

operation	effect
	address base initially set to zero
set address base to A 00000 0600 0000 aaaaa	address base at A_0
print I in octal 00000 4400 0000 iiiii	print two-dimensional array A_I in octal format
set address base to J 00000 0600 0000 jjjjj	address base at $A_{J,0}$
print M in octal, one word per line, with tag 00000 4150 0000 mmmmm	print block $A_{J,M}$ in the format one word per line, octal with tag
set address base back one level 00000 0700 0000 00001	address base at A_0
print K in decimal 00000 4430 0000 kkkkk	print two-dimensional array A_K in decimal format
set address base to N 00000 0600 0000 nnnnn	address base at $A_{N,0}$

operation	effect
print from R, P words at Q, in octal ppppp 4100 qqqq rrrrr	print words $A_{N,R,Q}$, ... , $A_{N,R,Q+P}$ in octal format
set address base to zero 00000 0700 0000 00000	return to initial address base setting

Note that the 0^{th} element of a block is denoted by 77777, and negative element addresses are specified in one's complement form.

The address base manipulation control word forms are as follows:

00000 0600 0000 fffff If address base is set to zero, set address base down to F. If address base is set down to A, set address base down one level to A_F.

00000 0700 0000 fffff Set address base back F levels. If F=0, set address base back to zero.

SPIREL execution of these control words causes no monitoring on the printer, but monitoring of SPIREL operations performed with the address base set to other than zero reflects the successive levels of address base settings in effect.

Print Symbol and Value Tables

As explained in the Genie Notes, system elements may have names which correspond to single word storage addresses and codeword addresses for arrays. When loaded under SPIREL control, such elements have their names and descriptive parameters stored in the SPIREL system vector *113, the Symbol Table (ST). The parallel SPIREL system vector *122, the Value Table (VT), contains the corresponding single word or primary codeword. The extent of the currently active ST-VT entries is maintained within the SPIREL system. These tables may be printed with basic SPIREL control words, but a special printing format appropriate to the contents of the tables is provided when the following control word is used:

nnnnn 0500 rrrr 00000

N words of ST and VT are printed, beginning at ST_R and VT_R. If N and R are empty, all of the currently active ST-VT entries are printed. The printing format used is discussed in a separate section.

Insert and Delete Words in Blocks

The insert operation provides a facility for lengthening and shortening blocks labelled by codewords. The insert control word form is

nnnnn 1150 rrrr fffff

The R^{th} word of the block labelled by codeword at F (counting from one if F contains no B-modifications, from the initial index otherwise) is addressed, and N words are inserted at that point in the block. N is interpreted in one's complement arithmetic. If N>0, N words containing zeros are inserted beginning at the R^{th} word, and the former R^{th} word becomes the $R+N^{th}$ word; the length of the block is increased by N. If N<0, N words are deleted beginning at the R^{th} word, and former $R+N^{th}$ word becomes the R^{th} word; the length of the block is decreased by N. If R is empty, N words are added to the end of the block. If N is empty, the R^{th} word and all following are deleted.

The insert operation as carried out by SPIREL requires that space for the new form of the block be available in the memory while the old form still exists. If STEX is active, the space occupied by the old form is freed when the new form is complete. If STEX is active and words are deleted from a block of codewords, the space occupied by the arrays whose codewords are deleted is freed.

Change Initial Index

The initial index of a block whose codeword contains B-modifications may be set by the control word

nnnnn 1160 0000 fffff

The initial index of the block labelled by the codeword at F is set to N, specified in ones complement form. After execution of this control word, the first word of the block is addressed as F_N. An initial index of zero is designated by N=77777.

Inactivate Storage

The inactivate operation may be applied to any array in the STEX domain by using the control word

00000 1170 0000 fffff

All storage for the array labelled by codeword at F is freed, and F is cleared. This operation is not meaningful unless STEX is active.

Monitor

If the STEX storage control system is active, blocks in the STEX domain are subject to being physically moved when it is necessary to concentrate free space. At the console, the user may wish to obtain information about the location of a particular block or the number of words of free storage. The control word

00000 3110 rrrr fffff

causes SPIREL monitoring to occur if SL 14 is off. R is interpreted as in standard address specification as the word in the block labelled by the codeword at F for which monitoring is desired. No SPIREL operation is performed on the block.

Obtain Date and Time

The date and time of day are available in the computer. SPIREL will format this information (14 positions in length) for printing when the control word

00000 4300 rrrr 00131

is executed. The next line actually printed will contain the date and time beginning at print position R. SPIREL will format and print the date and time beginning at print position R when the control word

00000 4310 rrrr 00131

is executed. The print positions are numbered 1-108 (decimal) from left to right across the pate. In both cases, if R is empty, the print position is set by SPIREL to 48 so that the date and time will appear at the right side of a page $8\frac{1}{2}$ inches wide.

Execute Control Word Sequence

A block may contain SPIREL control words. The execute control word sequence operation, when applied to the block, will cause SPIREL to interpret words addressed as control words and carry out the specified operations in the sequence given. The control word forms which instruct SPIREL to execute a sequence of control words are as follows:

nnnnn 6000 0000 fffff Execute N control words stored in memory from location F to location F+N-1.

nnnnn 6100 rrrr fffff Execute control words in the block labelled by codeword at F, where N and R are specified according to the standard rules for address specification.

Load Symbolic Cross References

Programs which are written to refer to named quantities, in particular Genie generated programs, require internal cross references to single words and codewords in the Value Table (VT, SPIREL system vector *122), parallel to the name of the quantity in the Symbol Table (ST, SPIREL system vector *113). The load symbolic cross references control word is

nnnnn 7100 0000 fffff

where F is the codeword address of the program into which N references are to be loaded. N units of information are read from paper tape. The format of this input and the use of symbolic cross references are explained in the Genie Notes, section on Symbolic Cross References.

IV.6. Recursive Application of SPIREL

In general, SPIREL control words with x=1 cause the specified operation to be applied to the block labelled by codeword at F. If meaningful, x=1 may be replaced by x=4 and the operation will be applied through the array labelled by codeword at F. This recursive application is accomplished by the use of SPIREL by SPIREL. In other words, when the SPIREL program *126 (XCWD) encounters a control word C with x=4 (except in the case of read, w=0), and F labels a block of codewords, the address base is set down to F, SPIREL is applied to the first N blocks on the next level with N control words C' in which N'=R and R'=0, and the address is set back up one level. If a control word with x=4 is applied to a block which does not contain codewords, the behavior of *126 is as if x=1, and the recursion is terminated. Thus the depth of the recursion is determined by the structure or depth of the array addressed.

As an example, consider the control word to print in decimal

00003 4430 0002 00200

where the array *200 is a standard data matrix. Since the block labelled by the codeword at 200 contains codewords, these control words are generated and delivered for SPIREL execution:

00000 0600 0000 00200
to set address base to 200

00002 4430 0000 00001
to print the first two words of row 1

00002 4430 0000 00002
to print the first two words of row 2

00002 4430 0000 00003
to print the first two words of row 3

00000 0700 0000 00001
to set address base back one level, to zero.

If row 2 in the array *200 had contained codewords for 4 blocks of data, the control word

00002 4430 0000 00002

in the above sequence would have caused further generation of the control words:

00000 0600 0000 00002
to set address base to 200_2

00000 4430 0000 00001
to print all of block $200_{2,1}$

00000 4430 0000 00002
to print all of block $200_{2,2}$

00000 0700 0000 00001
to set address base back one level, to 200

IV.7. Summary of Control Words

The chart on the next page is a reference for the SPIREL operations on the basis of w and y in the control word.

y \ w	0 read	1 correct	2 tagset	3 execute	4 print	5 punch	6	7
0	octal	octal	class 0	execute program	octal, 4 words/ line	octal	execute control word sequence	load symbolic cross refs
1	hexad	hexad	class 1	on-line control word print only	hexad, 108 chars/ line	hexad		
2	zeroes	zeroes	class 2	activate STEX	octal program format, 1 word/line	zeroes (control words only)		
3	decimal	decimal	all classes		decimal, 5 words/ line			
4	hexad + tag + check-sum	hexad + tag + check-sum	class 4		octal, 3 words/ line $8\frac{1}{2}$" wide	hexad + tag + check-sum		
5		insert/ delete space	class 5		octal + tag, 1 word/line			
6		change initial index			decimal + tag, 1 word/line			
7		free storage			decimal, 3 words/ line $8\frac{1}{2}$" wide			

SUMMARY OF SPIREL CONTROL WORD OPERATIONS.

V. Use of SPIREL

V.1. Input Paper Tape Formats

The SPIREL read control words designate the read mode to be employed in reading the pertinent data from paper tape. For each read mode there is an appropriate punch format for the data on paper tape.

The octal format (y=0) prescribes that each word consist of exactly 18 octal digits and that each word be preceded by a "spill character", usually a 'carriage return' punch. Octal tapes may be punched manually on the flexowriter. Also, a punch control word with y=0 produces octal format on paper tape, but this is inefficient use of paper tape since only three channels are utilized.

The hexad format (y=1) prescribes that each word consist of exactly 9 hexads and that each be preceded by a "spill character", usually a 'carriage return' or 'tab' punch. The hexad format utilizes all six data channels on paper tape and is equivalent to the octal format at twice the density. Hexad tapes are usually not punched manually on the flexowriter but are easily produced through SPIREL with a punch control word in which y=1.

The hexad with tag and checksum format (y=4) is produced only by a SPIREL punch control word with y=4. For example, output tapes from the assembly program and the compiler are in this form. The advantages of this format over the plain hexad format are implied by the name:

- Tags on words are represented on paper tape and reproduced when the tape is read.
- A sum over all words is formed as the tape is punched and represented on the tape. This sum is recomputed when the tape is read and the computed sum is compared to that punched on the tape. This provides a check on both punching and reading.

The format for one word consists of 9 hexads and one tag triad per word, where tag representations are

1 for tag 1
2 for tag 2
3 for tag 3
4 for no tag

The words with tag are not separated by any "spill characters" but are immediately adjacent to each other. A set of words punched with a single punch control word is preceded by a single "spill character" and followed by the checksum, one hexad word which is the fixed point sum of right half-word plus left half-word plus tag representation for all words in the set.

The decimal format (y=3) for a single word depends on the internal representation desired for the number. Tapes in decimal format may not be punched by SPIREL, but this is the format most frequently utilized for manually prepared data input tapes. A set of decimal words to be read due to the execution of a single read control word with y=3 is begun with a 'lower case' punch. Spaces and case punches are then ignored, and a punch other than one of

0 1 2 3 4 5 6 7 8 9 . + - e f t *

terminates a number which is being read. The character punched after each number to terminate it is most frequently a 'carriage return' punch. In the particular formats which follow, the letter d stands for a decimal digit, one of

0 1 2 3 4 5 6 7 8 9

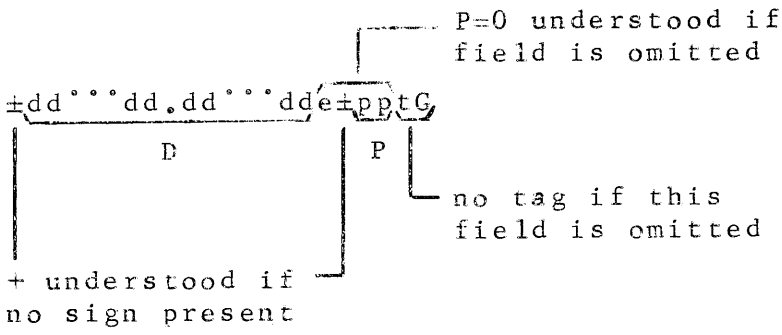
The punches 'e' and '*' may be used interchangeably. If a decimal point is punched in representing a number, it may begin or end the number. The particular decimal formats are as follows:

integer ±D of no more than 14 decimal digits, with tag G=1,2,3,4. G=1,2,3 causes the number to be stored with tag 1,2,3. G=4 causes the number to be stored without changing the tag in memory.

±dd···dddtG
D
no tag if this field omitted
+ understood if no sign present

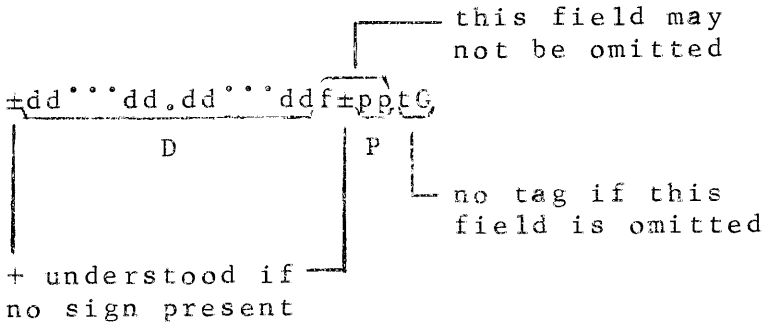
The decimal point is assumed to be to the right of the least significant digit punched and at the right end of the machine word. Integer fixed point arithmetic is to be employed.

floating point $\pm D \times 10^{\pm P}$, in absolute value between 10^{-78} and 10^{74}, with tag G=1,2,3,4



Decimal digits in D beyond the 14th are ignored. Floating point arithmetic is to be employed. The floating point form is recognized only if D contains a decimal point or if the field e±pp (or *±pp) appears.

fixed point fraction $\pm D \times 10^{\pm P}$ in absolute value $\geq 2^{47}$ and <1, with tag G=1,2,3,4.



Decimal digits in D beyond the 14th are ignored. The representation in the machine assumes that the decimal point is at the left end of the mantissa. Fixed point arithmetic is to be employed.

V.2. Tracing

Tracing is a means of observing the execution of instructions. This facility is provided in the SPIREL system. The instructions to be traced must bear a tag 3. Mode light 3 and trapping lights 6 and 9 must be turned on prior to a run in which tracing is to occur; this is the normal ML, TL configuration when SPIREL is initially loaded. The trace is provided by the SPIREL system program *13 (TRACE).

The printed trace output for each tagged instruction executed is given in eight fields as follows:

(CC): address of the instruction relative to the first word of the last program tagged with a tagset control word

(CC'): address of the next instruction to be executed relative to the first word of the last program tagged with a tagset control word, if not the next in sequence; zero otherwise

(ATR'): contents of the arithmetic tag register (1,2,or 3) after execution of the instruction; blank if the tag is zero

(M'): the final address formed as a result of decoding field 4 of the instruction

(I): the instruction executed

(S): the contents of S as a result of execution of field 4 of the instruction, before the operation in field 2 is carried out

(U'): contents of U after execution of the instruction

(R'): contents of R after execution of the instruction

The trace procedure distrubs no machine conditions other than the contents of P2 and S. If the contents of either of these registers is to be preserved from one instruction to the next, neither may be traced. For example, transfers to SPIREL system programs *136 (SAVE) and *137 (UNSAVE) set P2 for later retrieval by the programs being transferred to; these instructions may not be traced.

An order to be executed in the repeat mode may be tagged for tracing only if the order which causes entry into the repeat mode is also traced and is of the form

MLN

or SBi,ERM

or ABi,ERM

and the instruction executed immediately after the repeated instruction is also traced. The trace output for the traced repeated

instruction consists of one printed line in which (CC), (I), and (S) pertain to the first execution and (CC'), (ATR'), (M'), (U'), and (R') pertain to the last execution.

Manual controls may be applied to the trace procedure from the console by use of Mode Lights 14 and 15:

If ML15 is on, the tag will be erased from each instruction after it is traced, so that further execution will produce no trace output. If ML15 is off, the tag is retained.

If ML14 is on, a tagged instruction will produce trace output only if a branch of control occurs, that is if column (CC') of the trace output is not zero.

If ML14 and ML15 are both on, a tagged instruction will produce trace output only if a branch of control occurs, and the tag will then be erased from the traced instruction.

A trace program which provides printed output in decimal and one which operates on instructions bearing a tag 2 are available. These are not components of the SPIREL system but may be used in conjunction with the system.

V.3. Diagnostic Dump

If a program stops unexpectedly, the contents of machine registers at the time of the stop may be printed with mnemonic register labels by using the diagnostic dump, SPIREL system program *120, with entry prefix in machine locations 20000-20007 (octal). The procedure for obtaining this output is as follows:

- Record, mentally or otherwise, the contents of CC as displayed on the console if the instruction at which the stop occurred is of interest.
- Type 20000 (octal) into CC and FETCH to pass control to the diagnostic dump routine.
- A programmed halt occurs within the diagnostic dump routine. Type the recorded (CC) into U and CONTINUE; or simply CONTINUE if (CC) was not recorded.
- Diagnostic dump output is provided on the printer. If nothing was typed into U at the halt, CC and I at the time of the stop are indicated as containing 0.

• A programmed halt and transfer to location 20 occurs. This is the standard halt for external communication with SPIREL. All registers except U, CC, P2, S, and I are restored to their settings at the time of the stop.

V.4. High-Speed Memory Dump

If a program fails in such a way that the SPIREL system cannot be reached, a printed record of the memory configuration at the time of the failure is occasionally of assistance in debugging. For this purpose a self-loading High-Speed Dump tape is available at the console.

To load the tape, position the Dump tape in the reader, depress RESET, then LOAD. Do not CLEAR. When the Dump program is loaded, a halt occurs with

(I): 00 00000 00 0000 17561

Pushing continue causes dumping of the contents of memory, beginning at location 10.

The dump output is printed with four full words per line and the address of the first word at the left of the line. Each full word is split into five fields, corresponding to the fields of the machine instruction. If a word is tagged, an a, b, or c (corresponding to tag 1, tag 2, or tag 3) is printed immediately to the right of the tagged word.

No all-zero lines are printed; if a block of words spanning one line or more contains only untagged zeros, the paper is advanced one line to indicate the ellipsis.

Dumping terminates with the halt

(I): 77 00000 77 4001 17666

No registers are restored.

To begin dumping from some location other than 10, turn on the Tag 1 indicator light while the tape is loading. In this case, the halt after loading is

(I): 01 00001 20 4000 17561

At this halt, typing the desired starting location into B1 and pushing CONTINUE causes dumping to begin at (B1).

V.5. Error Halts in SPIREL

Programmed error halts within the SPIREL system and their significances are as follows:

(I): 67 77777 40 4001 XXXXX

The punched checksum for the block just read from paper tape did not agree with that computed while reading. Raising and lowering the F3 switch at the console exits the reading program and SPIREL *126.

(I): 41 00010 47 4001 XXXXX

Sufficient space is not available in memory for execution of the "read" control word just interpreted by SPIREL. CONTINUE to exit SPIREL *126.

(I): 41 00010 33 4001 XXXXX

Sufficient space is not available in memory for execution of the "chain" control word just interpreted by SPIREL. CONTINUE to exit SPIREL *126.

(I): 41 00010 11 4001 XXXXX

Sufficient space is not available in memory for execution of the "insert" control word just interpreted by SPIREL. CONTINUE to exit SPIREL *126.

(I): 77 00000 00 4001 XXXXX

An improper decimal number has just been read from paper tape, out of the range permitted for the format used. CONTINUE causes exit from the reading program and SPIREL *126.

V.6. Symbol Table-Value Table Print Format

The SPIREL control word

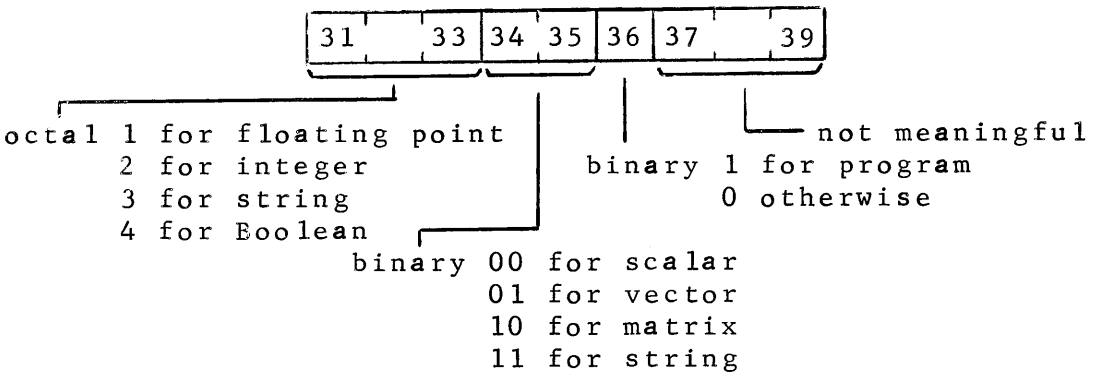
nnnnn 0500 rrrr 00000

provides parallel printing of corresponding Symbol Table (ST) and Value Table (VT) entries. These "tables" are SPIREL system standard vectors *113 and *122 respectively. The output is in seven fields as follows:

(1) relative address in vector, i.e., index

(2) symbol from bits 1-30 of ST entry

(3) descriptive parameters in octal from bits 31-39 of ST entry



(4) address field from ST entry (not used by SPIREL)

(5) tag on ST entry

(6) VT entry: value associated with name in ST entry if it is a single-word quantity; codeword for array associated with name in ST entry, empty if the array does not currently exit

(7) tag on VT entry

The Symbol Table and Value Table are most commonly used when a system contains named Genie-generated programs and the named data referred to in such programs. This usage is fully described in the Notes on the Genie Compiler.

V.7. SPIREL System on Magnetic Tape

Several copies of the SPIREL system are located on the MT System magnetic tape. When one of these copies is read into the memory a halt occurs in the SPIREL external communications routine. The Mode and Trapping Lights are set to permit tracing of tagged instructions. Loading of "private" blocks will begin at about location 3700 (octal).

V.8. SPIREL System on Paper Tape

The SPIREL system is available on paper tape. When the tape is readied in the reader, the console CLEAR and LOAD operations should be performed. The tape is read, finishing successfully with a page restore and printing of the word "SPIREL". A halt occurs in

the SPIREL external communications routine. The Mode and Trapping Lights are set to permit tracing of tagged instructions. Loading of "private" blocks will begin at about location 3700 (octal).

VI. Storage Control

VI.1. Linear Consumption by TAKE

The simple storage control program which is initially utilized in the SPIREL system is called TAKE, program *135. TAKE operates on the principle of linear consumption of memory. A pointer to the first inactive word of storage, address L, is maintained. A request for M words is satisfied by an allocation of M words at L, and L is incremented by M. This is an irreversible procedure in that space, once allocated, may not be reclaimed for use in later allocations. L is stored in the address field of location 100 (octal) in the SPIREL system.

TAKE may be utilized "privately" to obtain blocks of memory in a way compatible with allocation by the SPIREL system. On entry to TAKE, program *135, (B2)=number of words desired. On exit (B1)=address of first word of block allocated. If the request for space cannot be satisfied, (B1)=0 on exit.

VI.2. Activation of STEX and its Domain

The SPIREL Storage Exchange program STEX, *154, is activated by the control word

00000 3120 0000 00135

In fact, program *154 becomes program *135 and TAKE is discarded. If L is the address of the first word of inactive storage at the time STEX is activated, the STEX domain is [L, 17577], all inactive storage. STEX cannot be deactivated for recourse to the TAKE system. STEX provides optimal use of storage because

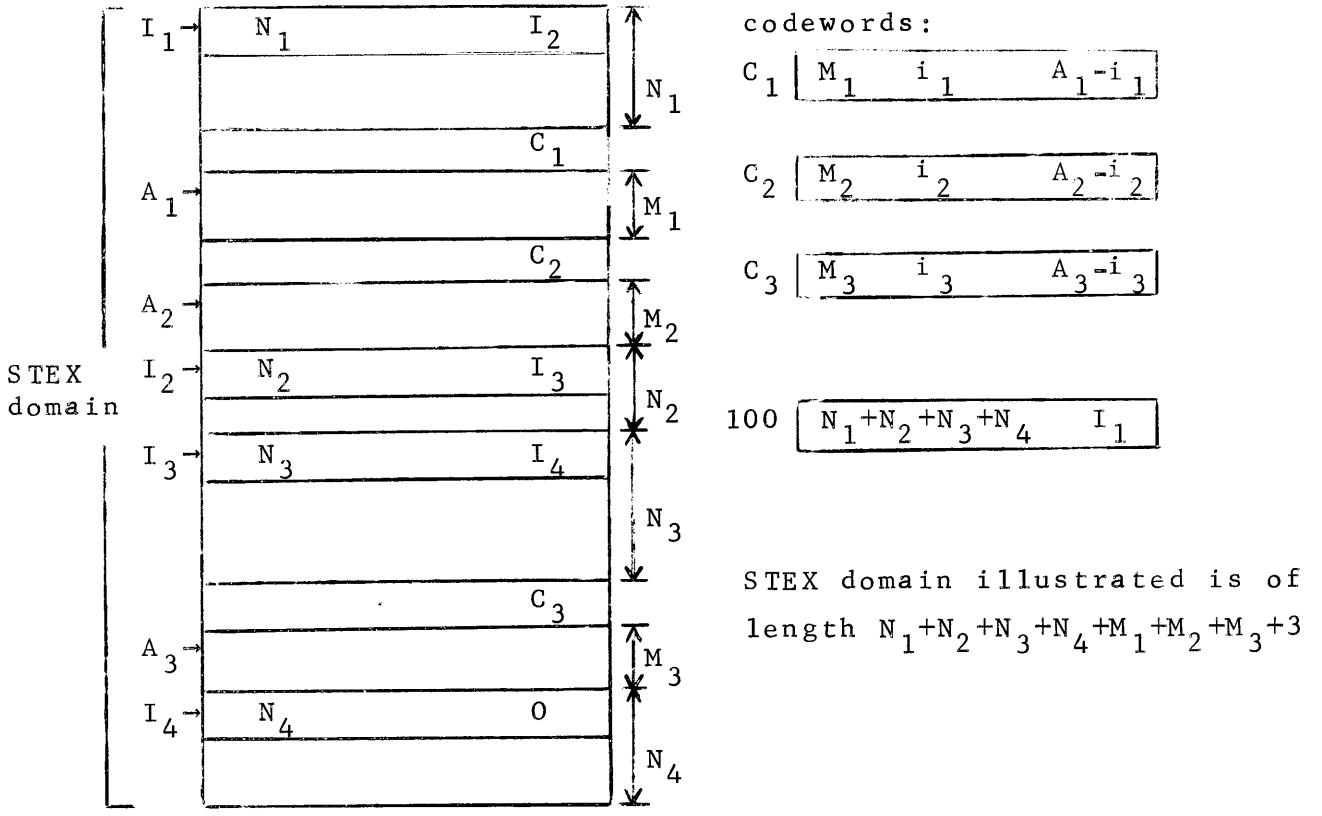
- blocks may be freed explicitly, making such space as becomes logically unnecessary available for reassignment to blocks logically required;
- blocks whose codewords are re-used to label new blocks are automatically freed for re-use;
- if the total free space in memory is sufficient to satisfy a request for a block but no single space is large enough, free space is automatically concentrated and the allocation is made.

Any block which is subject to being freed, either explicitly or implicitly, after STEX activation, must be in the STEX domain, i.e., must be created after STEX is activated.

VI.3. Memory Configuration Generated by STEX

The domain of STEX is divided into active and inactive areas. Each area is further segmented into blocks, not necessarily adjacent. Each active block is labelled by a codeword. The first word of an active block is not a part of the block from the user's point of view; it is called the back-reference for the block and contains in its address field the codeword address of the block. The inactive blocks are chained from location 100, which contains the total number of inactive locations in the memory in its first 15 bits and the address of the first inactive location in its last 15 bits. Each inactive block contains in its first word the length of the block it heads and the address of the first word of the next block, the last inactive block being distinguished by a zero address field, thus ending the chain. The first word of an inactive block is inactive in the sense that it may be activated just as any other word in the block.

An illustration of the STEX memory configuration is given below:



In general, the length of the STEX domain is given by

$$K = \sum_{i=1}^{n} N_i + \sum_{j=1}^{m} M_j + m$$

where the domain is divided into n inactive blocks and m active blocks. Note that K is a constant, determined at the time STEX is activated, and K=17600-L, where L is the address of the first word of inactive storage at the time STEX is activated.

The codewords that address the active blocks may be located anywhere in the memory, but the blocks of every array must lie wholly inside or outside the domain of STEX.

If it is necessary to move a codeword for an active block, the back-reference for the block must be appropriately altered. For instance, when the block addressed through codeword A is activated the back-reference for the block contains the address A; if (A) is stored at B, back-reference for the block must be changed to B.

STEX offers many advantages for data storage control, but programs may also be loaded into the STEX domain with a few restrictions. Since pathfinder settings are absolute and return is not made through codewords, programs should not be moved in a memory reorganization which STEX may have to perform. This possibility is eliminated if programs are loaded before any space is taken for data that may ever be inactivated. This rule should always be followed.

VI.4. Use of STEX

Once activated, STEX exists as program *135 and may be used directly by the coder. The entry parameters are

(B1)=codeword address of block on which STEX is to operate,

(B2)=length of block.

STEX first tests the word addressed by (B1). If this codeword is not null, the storage addressed through this codeword to all sublevels is inactivated by STEX and all codewords are made null. If the codeword is null, no inactivation occurs. Then (B2) is tested. If (B2)≠0, a block of storage of length (B2) is

activated with back-reference to the address (B1), the codeword at (B1) remains null, and then (B1) is set for exit to the FWA of the block activated. Hence, to take N locations to be addressed through codeword C, set (B1)=C, (B2)=N and enter STEX. All space formerly occupied by array C will be inactivated and all associated codewords cleared. Exit will be made with (B1)=FWA of a new block to be addressed through C and (B2) unchanged.

To simply inactivate memory addressed through C, enter STEX with (B1)=C and (B2)=0. If (B1)=0 on entry, a memory reorginization is performed (see below) and no space is taken.

If the inactive area is not sufficiently large to meet a request for space, exit is made with (B1)=0. If the total inactive storage will accomodate a request for space, STEX will activate a block by one of the three following means:

(1) space is activated from within an existing inactive block;

(2) if (1) fails, adjacent inactive blocks are combined and (1) is tried again;

(3) if (2) fails, all active memory is re-written to the low end of the domain, leaving one inactive block at the high end, thus forcing (1) to succeed.

Alternative (3) is called reorganization; if this occurs and SL14 is off, the message REORGANIZATION is printed.

When STEX is used directly, the coder must generate his own codewords. The alternative of taking space with a "read" control word provides generation of codewords for the coder.

VII. SPIREL System Components

Any component which is of use to the individual programmer is denoted by • in the margin next to its name.

Since programs *135(TAKE), *136(SAVE), and *137(UNSAVE) are necessary components of the SPIREL system, they are not included in the lists of supporting routines.

VII.1. Vectors and Print Matrix

*113, Symbol Table, ST

Length: 100 (octal)

Function: This is a standard B1-modified vector. Each entry contains the name and descriptive parameter for an item in the total system being run, an item which is identified symbolically rather than by its address or codeword address. The parallel entry in the Value Table, *122, contains the item or codeword corresponding to the item name in the Symbol Table. The index of the last active entry in the Symbol Table is dynamically maintained at location 117.

• *116, Print Matrix, PM

Length: 200 (octal)

Function: This block is not B-modified. The address of the first word of *116 is used as the address field in all SPIREL print orders, except in tracing. The print matrix is always cleared immediately after SPIREL-controlled printing.

*122, Value Table, VT

Length: 100 (octal)

Function: This is a standard B1-modified vector. Each entry contains the value of or the codeword for an item in the total system being run, an item which is identified symbolically rather than by address or codeword address. The parallel entry in the Symbol Table, *113, contains the name and descriptive parameter for the item. The index of the last active entry in the Value Table is dynamically maintained at location 117.

*125, Base Address Vector, ADDR

Length: 6

Function: This block is not B-modified. It is used by SPIREL to dynamically maintain a record of all levels through which the

base address had been set down and to compute effective addresses from those specified in control words.

VII.2. Programs

*13, Trace, TRACE

Length: 251 (octal)

Function: The SPIREL trace program receives control through hardware trapping due to tag 3 on instructions, both before and after execution of the instruction. Information for each line of trace output is derived, formatted, and printed by this program.

Registers Not Preserved: P2,S

Supporting Routines: *127(SETPM)

*110, Print Control Word, HDPR

Length: 55 (octal)

Function: This program is used in the SPIREL system for on-line control word monitoring when SL14 is off.

Supporting Routines: SETPM(*127)

*111, Process Matrix, MATRX

Length: 135 (octal)

Function: This program is used in the recursive application of SPIREL as explained elsewhere.

Supporting Routines: all SPIREL components, but only those necessary to perform the specified operation on any particular utilization; see section on SPIREL Component Linkages.

*120, Diagnostic Dump, DIADMP

Length: 57 (octal)

Function: This program is used in the SPIREL system when control is passed to location 20000 (octal). The diagnostic dump formats and prints the contents of the fast registers as explained in the section of Use of SPIREL.

*126, Execute Control Word, XCWD

Length: 276 (octal)

Function: This program is the nucleus of the SPIREL system. It interprets control words and may use other system programs to carry out specified operations. External communication to XCWD from paper tape and from the console is provided within the system

and is explained elsewhere. For internal communication, control should be passed to the second word of *126 if the SPIREL operation specified is to be performed on a named item; otherwise control is given to the first word of *126.

Input: (T7)=SPIREL control word to be executed.

(T4)=5 left-adjusted printer hexads (with '25' fill if necessary) for name of item to be operated on if control is given to *126 at the second word; in this case F in the control word in T7 is empty.

Registers Not Preserved: none (and SPIREL cannot operate on fast registers)

Supporting Routines: all SPIREL components, but only those necessary to perform specified the operation on any particular utilization; see section on SPIREL Component Linkages.

- *127, Set Up Print Matrix, SETPM

Length: 75 (octal)

Function: This program will format a single word into the print matrix for printing at a specified position on a line if appropriately instructed. It will print the contents of the print matrix and clear the print matrix if appropriately instructed. Common usage of SETPM for the printing of a single line consists of an entry for each word of information to be printed and an entry to have the collection printed and the print matrix cleared. Thus, SETPM provides a facility for composition and output of lines on the printer.

Input: (T7)=information to be set up in print matrix, if any

(B1)=parameter which controls operation of SETPM

(B3)=print position, 1-108 (decimal) at which field set-up should begin, if relevant.

SL15 on causes leading zeros in a hexad field to be deleted; leading zeros in numeric fields are automatically deleted unless SL15 is manually locked off.

Operation: on the basis of (B1) on entry:

(B1)<0, octal format of last 5 triads of (T7) at print position (B3), and increment of (B) by 5

(B1)=0, octal format of (T7) in 18-position field at print position (B3), and advance of (B3) by 18
(B1)=1, hexad format of (T7) in 9-position field at print position (B3), and increment of (B3) by 9
(B1)=2, print and clear PM, (B3) set to one and (T7) meaningless
(B1)=3, decimal format of (T7) in 18-position field at print position (B3), and increment of (B3) by 18

Registers Not Preserved: none

Supporting Routines: *155 (BINDC) when decimal formatting is used.

*130, Find Symbolic Name, SMNAM

Length: 26 (octal)

Function: This program is used in the SPIREL system for operations which are performed on items with symbolic names.

Supporting Routines: *176 (TLU)

*131, Print Date and Time, DATIME

Length: 14 (octal)

Function: This program reads the digital clock through *132 (CLOCK), sets up the 14-character date and time in the print matrix, and prints the contents of the print matrix if requested to do so. It is used by XCWD to perform "obtain date and time" operations; it may also be used directly.

Input: T7=00000 00p0 rrrr 00000,
where p=0 causes set-up only
p=1 causes set-up, print, and clear
r specifies the print position of the date
r=0 causes setup for 8 1/2 x 11" pages
(first character in print position 48)

Registers Not Preserved: T7

Supporting Routines: CLOCK(*132), SETPM (*127)

*132, Decode Clock, CLOCK

Length: 55 (octal)

Function: This program interrogates the digital clock and

calendar in the machine and translates the coded time and date into printer hexads. The time is based on a 24-hour clock. CLOCK is used by DATIME(*132).

Input: none

Output: Printer hexads in T5, T6. For example, if CLOCK were executed at 9:45 pm on May 17, 1964, the CLOCK output would be:

T5=05/17/64_
T6=17.45_ _ _ _

where _ denotes "space".

Registers Not Preserved: B2,T4,T5,T6,T7

Supporting Routines: none

*133, Punch Control Word, PCNTRL

Length: 15 (octal)

Function: This program is used in the SPIREL system for any punch operation executed with SL14 off.

Supporting Routines: none

*134, Set Up Word in Program Format in Print Matrix, FIELD

Length: 20 (octal)

Function: This program is used in the SPIREL system for the operation of printing in program format.

Supporting Routines: none

*135, Take Memory Space, TAKE

Length: 5

Function: This program performs linear irreversible storage allocation for blocks and maintains location 100 as a pointer to the first word of inactive storage.

Input: (B2)=length of block to be allocated.

Output: (B1)=address of first word of block allocated, 0 if space not available for specified allocation.

Registers Not Preserved: none

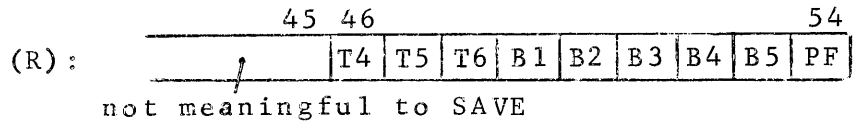
Supporting Routines: none

*136, Save Fast Registers, SAVE

Length: 41 (octal)

Function: This program uses the B6-list for storage of specified fast registers and a word denoting the registers so stored.

Input: (R), bits 46-54, to specify registers to be saved:



For each position in which the bit is 1, the corresponding register will be saved. The registers are saved on the B6-list in the order shown from right to left (i.e., PF, if specified, saved first), and the (R) is itself stored on the B6-list. Notice that (R)=-Z on entry causes all nine registers to be saved.

Use: Control should be passed to SAVE by a TRA (not a TSR) instruction so that the saving of (PF) is meaningful. SAVE returns via (P2) on entry, and the instruction

```
TRA     *136
```

must not be traced.

Registers Not Preserved: none

Supporting Routines: none

*137, Unsave Fast Registers, UNSAVE

Length: 41 (octal)

Function: This program complements SAVE. It obtains from the B6-list the (R) stored by the complementary execution of SAVE, restores all fast registers then saved, and decrements B6 appropriately.

Use: Control should be passed to UNSAVE by a TRA (not a TSR) instruction. UNSAVE returns via (P2) on entry, and the instruction

```
TRA     *137
```

must not be traced.

Registers Not Preserved: none

Supporting Routines: none

*140, Insert or Delete Space, DELETE

Length: 124 (octal)

Function: This program is used in the SPIREL system for the operation of inserting or deleting words in blocks.

Supporting Routines: none

*141, Change Initial Index, CHINDX

Length: 13 (octal)

Function: This program is used in the SPIREL system for the operation of changing the initial index of a block.

Supporting Routines: none

*142, Tagset, TAGSET

Length: 15 (octal)

Function: This program is used in the SPIREL system for the operation of tag setting.

Supporting Routines: none

*143, Take Chain Storage, CHN

Length: 13 (octal)

Function: This program is used by the SPIREL system for the operation of generating chain storage.

Supporting Routines: none

*144, Print, PRINT

Length: 70 (octal)

Function: This program is used in the SPIREL system for all print operations.

Supporting Routines:

*127(SETPM)	for all printing
*155(BINDC)	for decimal printing
*150(PRNWTG)	for printing with tags
*134(FIELD)	for printing in program format

*145, Punch, PUNCH

Length: 130 (octal)

Function: This program is used in the SPIREL system for all punch operations.

Supporting Routines: *133(PCNTRL) for punching tapes with control words (i.e., with SL14 off); *157(PUNCHK) for punching tapes in the hexad with tag and checksum format

*146, Execute Control Word Sequence, XCWSQ

Length: 15 (octal)

Function: This program is used in the SPIREL system for the operation of executing a control word sequence.

Supporting Routines: all SPIREL components, but only those necessary to perform the operations specified by the control words in the sequence being executed; see section of SPIREL Component Linkages.

*147, Load Symbolic Cross References, SYMRF

Length: 12 (octal)

Function: This program is used in the SPIREL system for the operation of loading symbolic cross-references.

Supporting Routines: *176(TLU)

*150, Set Up Tag in Print Matrix, PRNWTG

Length: 16 (octal)

Function: This program is used in the SPIREL system for the operation of printing with tags.

Supporting Routines: none

*151, Print Symbol and Value Tables, PRSYM

Length: 43 (octal)

Function: This program is used in the SPIREL system for the printing of ST and VT in the special format described elsewhere.

Supporting Routines: *127(SETPM)

*152, Conversion of Powers of Ten, PWRTN

Length: 41 (octal)

Function: This program is used in the SPIREL system for reading of decimal numbers from paper tape. Given the floating point number and the integer Q, this program computes the floating point number $N=P\times10^{Q}$.

Input: (T5)=signed floating point number P
(B2)=integer Q in one's complement form, less than 75 (decimal) in absolute value

Output: $(T5)=P\times10^{Q}$, (PF)=0 if $|Q|<75$
(T5)=0, (PF)=1 if $|Q|\geq75$

Registers Not Preserved: B1, T4

Supporting Routines: none

*153, Multiple Read Decimal, MRDDC

Length: 144 (octal)

Function: This program is used in the SPIREL system for the reading of data in decimal input formats from paper tape. It may be used directly to read decimal numbers, convert them as explained elsewhere, and store them.

Input: (B1)=address at which to begin storing the numbers read.

(B2)=number of numbers to read.

Output: (B1) same as on input.

(B2)=0

Error Halt: The SPIREL error halt

(I): 77 00000 00 4001 XXXXX

occurs if an improper decimal number is read, one which is out of the range permitted for the format used. CONTINUE causes exit from MRDDC with no further reading.

Registers Not Preserved: none

Supporting Routines: *152 (PWRTN)

• *154, Storage Exchange, STEX

Length: 213 (octal)

Function: This program must be activated with the SPIREL control word

00000 3120 0000 00135

before it may be used. It then operates as program *135 and must be used as such. The function of STEX is explained in detail in the section on Storage Control.

Input: (B1)=codeword address of array or block which is to be freed or for which space is to be allocated; 0 if reorganization is desired.

(B2)=length of block to be allocated; 0 if no space is to be allocated.

Output: (B1)=address of first word of block allocated; 0 if allocation requested and insufficient space available; same as on entry if no allocation is requested.

(B2) same as on entry.

Registers Not Preserved: none

Supporting Routines: none

- *155, Binary to Decimal Conversion, BINDC

Length: 102 (octal)

Function: This program is used in the SPIREL system for conversion of a number from its internal binary representation to a decimal representation in printer hexads. It may be used directly for the same purpose.

Input: (T4)=numbers to be converted, fixed point integer if exponent empty, floating point otherwise.

Output: (T4), (T5)=number in decimal printer hexad form, 18 hexads:

floating point

±d.ddddddddddde±dd

12 decimal digits; exponent, 2 decimal digits

fixed point integer

bb···bb±dd···dd

1-16 blanks; 1-15 decimal digits

Registers Not Preserved: T6,T7,B1,B2,B3

Supporting Routines: none

- *156, Read with Checksum, RDCHK

Length: 35 (octal)

Function: This program is used in the SPIREL system for reading tapes in the hexad with tag and checksum format explained in the section of Use of SPIREL. It may be used for the same purpose by an individual.

Input: (B1)=address at which to store first word read.
(B2)=number of words to read.

Error Halt: The SPIREL error halt
(I): 67 77777 40 4001 XXXXX
occurs if the checksum computed while reading does not agree with that read from paper tape. Raising and lowering the F3 switch at the console will cause exit from RDCHK.

Registers Not Preserved: T4,T5,T6,T7, B2,B3,B4 ((B2)=0 on exit)

Supporting Routines: none

Word of Caution: The SPIREL hexad with tag and checksum format prescribes that each sequence which is checksummed be preceded by an extraneous "spill character" (one hexad). RDCHK, in fact, expects this hexad, but it is not punched by *157, PUNCHK.

- *157, Punch with Checksum, PUNCHK

Length: 24 (octal)

Function: This program is used in the SPIREL System for punching tapes in the hexad with tag and checksum format when the "spill character" (one hexad) that precedes a checksummed sequence is provided by another component of SPIREL. It may be used for the same purpose by an individual.

Input: (B1)=address of first word to be punched.
(B2)=numbers of words to be punched.

Registers Not Preserved: T4,T5,T6,T7, B3,B4

Supporting Routines: none

- *176, Table Look-Up, TLU

Length: 23 (octal)

Function: This program is used in the SPIREL system to determine the index of the Symbol Table entry for a name. If the name "looked for" does not appear on the Symbol Table, TLU adds it and increments the current last active index at location 117 by one.

Input: (T4)=left adjusted 5 printer hexad representation of the name to be "looked for" on the Symbol Table, *113; the rest of (T4) is irrelevant to TLU.

Output: (B1)=index of ST entry for the name "looked for".
(PF)=1 if the entry was added to the active portion of the Symbol Table; 0 otherwise.

Registers Not Preserved: B1, PF

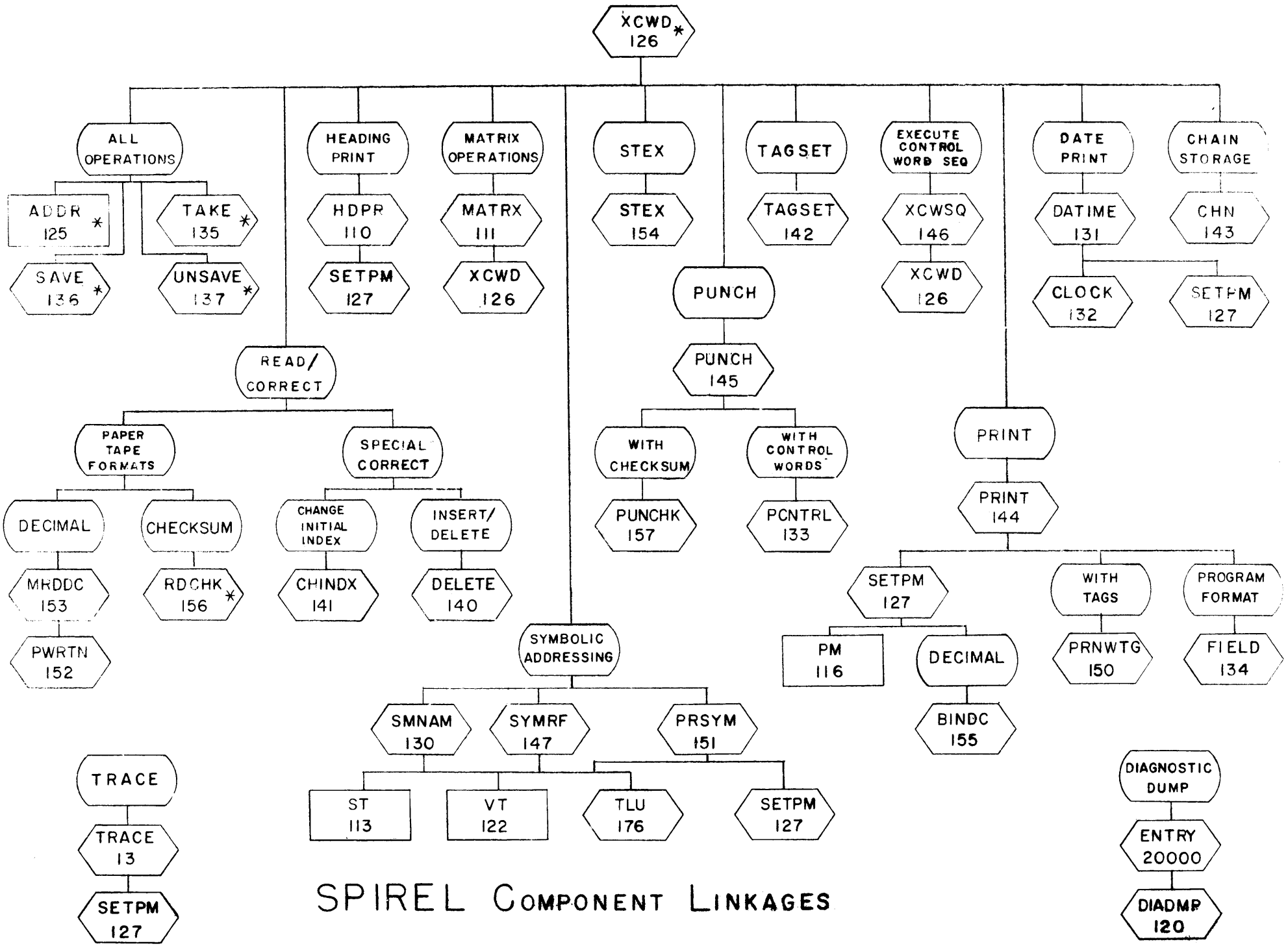
Supporting Routines: none

VI.3. Component Linkages

The chart on the next page shows linkages among SPIREL components. SPIREL operations are shown in ovals ( ), and each operation is linked to SPIREL components on which it may be dependent. Programs are denoted by hexagons ⟨ ⟩ and vectors by rectangles [ ]. The programs and vector required for a minimum SPIREL are starred (*) on the chart.

Reading the chart from bottom to top, programs and vectors are needed only to perform operations which appear in ovals above them (except for XCWD, which is always required).

For example, XCWD (*126) uses DATIME (*131) only when an "obtain date and time" control word (00000 4310 0000 00131) is executed; DATIME always uses CLOCK (*132) and SETPM (*127).



SPIREL Component Linkages

VIII. System Duplicator

VIII.1. Purpose of the Duplicator

When a system of programs has been debugged and extensive production running is contemplated or when memory space becomes critical, it may be advantageous to produce a self-loading paper tape bearing necessary SPIREL components, library routines used, and the coder's private system elements. The system duplicator MSPDUP, program *10, is designed for this purpose. This program is not itself a SPIREL component. MSPDUP has codeword address 10 (octal) and this will not conflict with normal codeword locations.

VIII.2. Use of the Duplicator

A dump tape is used to tell program *10 what system elements are to be punched. The dump tape consists of a series of dump words, each preceded by a 'carriage return' punch and consisting of 18 octal digits. The dump word forms are as follows:

00000 0100 0000 fffff for all of the block (program or vector) with codeword at F.

nnnnn 0000 0000 fffff for N words beginning at machine address F.

nnnnn 0120 rrrr fffff for a control word which will cause a block of N zeroes to be created with codeword at F and R in the dump word at the corresponding position in the codeword. If N in the dump word is empty, N and R will be obtained from the codeword at F at time of punching.

nnnnn 1100 rrrr fffff for N words beginning at the Rth element of the block with codeword at F. A "correct" control word is punched which is meaningful at later reading only if the block has been previously created.

A null dump word (18 '0' punches preceded by a 'carriage return') terminates the list of system components to be punched. Hexads on the dump tape after the null dump word will be reproduced onto the tape punched. Except for blocks of zeros, all components are punched in the hexad with tag and checksum format.

With the complete system from which components are to be punched to produce a self-loading system tape in the machine, execution of program *10 results in the following steps:

- A leader is punched. It contains the SPIREL loader and RDCHK, program *156.
- The programmed stop

  Z HTR CC

  occurs, and the dump tape should be readied in the reader.
- CONTINUEing causes the dump tape to be read and the specified system components to be punched, until the null dump word is read.
- The SPIREL system tail is punched.
- Any information on the dump tape beyond the null dump word is duplicated, hexad for hexad.

VIII.3. SPIREL Generation

A complete SPIREL system is produced by use of a dump tape as follows:

dump words		components
00000 0120 0000 00113		ST
00000 0120 0000 00116	†	PM
00000 0120 0000 00122		VT
00000 0100 0000 00013		TRACE
00000 0100 0000 00110		HDPR
00000 0100 0000 00111		MATRX
00010 0000 0000 20000		ENTRY
00000 0100 0000 00120		DIADMP
00000 0100 0000 00125	†	ADDR
00000 0100 0000 00126	†	XCWD
00000 0100 0000 00127	†	SETPM
00000 0100 0000 00130		SMNAM
00000 0100 0000 00131		DATIME
00000 0100 0000 00132		CLOCK
00000 0100 0000 00133		PCNTRL
00000 0100 0000 00134		FIELD
00000 0100 0000 00135	†	TAKE
00000 0100 0000 00136	†	SAVE
00000 0100 0000 00137	†	UNSAVE
00000 0100 0000 00140		DELETE
00000 0100 0000 00141		CHINDX
00000 0100 0000 00142		TAGSET
00000 0100 0000 00143		CHN
00000 0100 0000 00144		PRINT
00000 0100 0000 00145		PUNCH
00000 0100 0000 00146		XCWSQ
00000 0100 0000 00147		SYMRF

dump words	components
00000 0100 0000 00150	PRNWTG
00000 0100 0000 00151	PRSYM
00000 0100 0000 00152	PWRTN
00000 0100 0000 00153	MRDDC
00000 0100 0000 00154	STEX
00000 0100 0000 00155	BINDC
00000 0100 0000 00157	PUNCHK
00000 0100 0000 00176	TLU
00000 0000 0000 00000	Null word to end dump

†components which must be included in minimal SPIREL; in fact *116 and *127 may be deleted with corrections to MSPDUP.

For determination of a sufficient set of SPIREL components to satisfy to requirements of a private system, reference should be made to the diagram of SPIREL component linkages. If HDPR, program *110, is to be eliminated, all control words must be executed with SL14 on unless location 110 routes control immediately to the program using *110. Set

(110): 00000 0000 0001 00000

before execution of *10 and include the dump word

00001 0000 0000 00110

in the dump tape.

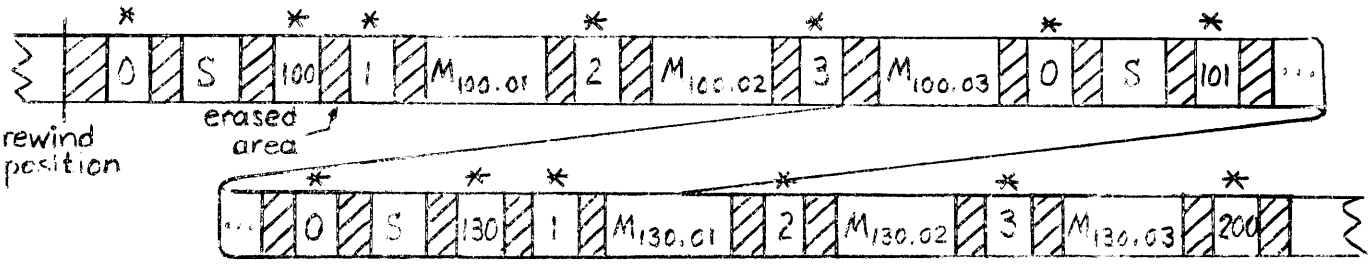
To alter the initial SPIREL loading address (normally octal 300) or the message printed upon completion of paper tape reading, or to delete *116 and *127, refer to the symbolic listing of MSPDUP in the UTILITY ROUTINES reference notebook.

IX. Magnetic Tape System

This system is designed to provide each user with an area in which he may store SPIREL - compatible systems and to allow retrieval of these systems with a minimum of knowledge about the status or position of the tape.

FORMAT

The tape will be pre-written in the following format:



where * denotes a mark as shown below it,

mark 0 is the system mark,

marks 100-130 are file marks,

marks 1-3 are block marks witnin files,

S is data, the tape system which loads above 17600,

M is data, a full-memory dump, initially a SPIREL and the pre-writing program.

The tape, then, is logically divided into files labelled 100 through 130 sequentially, each of which is subdivided into blocks labelled 1 through 3 sequentially. The memory following block mark j within file i is denoted $M_{i.j}$ on the preceding diagram.

POSITION

When expected to read or write, an S will be picked up off magnetic tape and display of the current position in the indicator lights will appear as

(IL): FFF01

meaning that the current position is at data area $M_{FFF.01}$. A halt occurs with an arrow displayed in U which points to the sense lights. At this time the sense lights should be set to

(SL): FFFBB

and when CONTINUE is pressed, the reading or writing will take place at data area $M_{FFF.BB}$. If FFF in IL is the file number desired, then

(SL): 000BB

is sufficient. If both FFF and BB in IL are as desired, (SL) null is sufficient. If FFF01 is desired, then

(SL): FFF00

is sufficient.

READ

Loading of the paper tape labelled READ will cause:

(i) reading of a copy of the system S from magnetic tape;

(ii) halt with arrow in U to obtain positioning information FFF.BB from SL;

(iii) reading of data area FFF.BB;

(iv) resetting of lights to status at the time of writing the block;

(v) setting of (B6)=17600;

(vi) transfer of control to the SPIREL sequence at 20, with execution of the control word specified at the time of writing the block.

If necessary, as many as eight attempts will be made to read each S and data area successfully, i.e., with no UME and with proper checksum. If the information cannot be read, a halt occurs with "NO" displayed in U; CONTINUE to try again.

WRITE

Clear the memory under program control (use paper tape labelled CLEAR). Next load the system to be written. Then turn off the "NOT WRITE" light on the transport, and loading of the paper tape labelled WRITE will cause:

(i) saving of lights;

(ii) reading of a copy of the system S from magnetic tape;

(iii) halt with arrow in U to obtain positioning information FFF.BB from SL;

(iv) halt with U null at which time a SPIREL control word may be typed into U (this will be executed after reading from tape the block about to be written); CONTINUE without typing if no control word is to be executed after reading;

(v) writing of the full memory at data area FFF.BB;

(vi) reading of the data area just written without storing to determine whether the data was written incorrectly or if the tape is defective in that block; if so, a halt occurs with (I):

77 00000 77 4000 17604

if CONTINUE is pushed, the program returns to step (iii) so that the write can be tried again or a different block of tape can be selected. A FETCH causes the program to continue to step (vii);

(vii) reading of the data area just written and entry into the sequence described in "read", starting at step (iii).

DO NOT write on the system tape except through this procedure.

The

A S S E M B L Y L A N G U A G E

for the

R I C E U N I V E R S I T Y C O M P U T E R

Programming development on the Rice University Computer has been supported by The National Science Foundation under grants G-7648 and G-17934. Construction of the computer was supported by The United States Atomic Energy Commission under contracts AT-(40-1)-1825 and AT-(40-1)-2572.

April, 1964

ASSEMBLY LANGUAGE

ASSEMBLY LANGUAGE

I. SYMBOLIC CODING

The absolute machine language of the Rice Computer is described in detail in the Rice Computer Manual. In practice, programs are not written in the absolute language of the computer but in a symbolic language. The symbolic language which provides notation for instructions, or commands, that correspond one-for-one with absolute machine instructions is called an assembly language. The program which translates assembly language into machine language is called an assembly program.

Use of the assembly language for the Rice Computer depends on a knowledge of the absolute machine instruction format, a familiarity with the registers of the computer, and a general acquaintance with the instruction repertoire -- all explained in the Rice Computer Manual. Two forms of the Rice Computer assembly language are available:

AP1, for independent use

AP2, for use within Genie programs

The corresponding assembly programs have the same names:

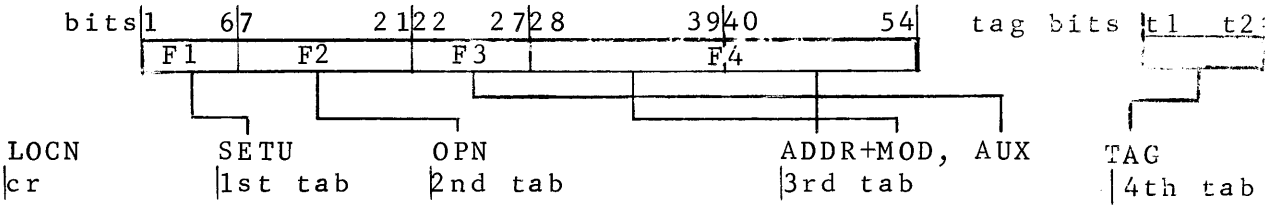
AP1, an independent assembly program within the AP1 PLACER system

AP2, a subset of the Genie compiler within the Genie PLACER system

The two assembly languages are very similar. The major distinction concerns octal and decimal numerals. In AP1, all numerical constants are assumed to be octal unless immediately preceded by the special symbol "d", meaning decimal. In AP2, all numerical constants are assumed to be decimal, except when octal form is indicated by a plus sign immediately preceding the octal number. In the following discussions, M stands for the final number formed in the last 15 bits of I (the instruction register) after all specified indirect addressing and B-modification has taken place; and if Q is any machine location, then (Q) stands for the contents of location Q.

I.1. Instruction Form

The general form of an AP1 or AP2 instruction and its correspondence to a machine-language instruction as explained in the Rice Computer Manual is



Here "cr" denotes "carriage return", and "tab" denotes "tabulate" on the flexowriter used for preparation of input to the assembly programs.

LOCN gives the symbolic form (if any) of the location of the instruction. SETU corresponds to Field 1: bring a "fast" register to U; then inflect (U). OPN corresponds to a Field 2 operation chosen from one of six classes (see section II). AUX corresponds to Field 3: alter a B-register, send (U) or (R) to a "fast" register, send the M portion of I to a B-register, or clear R. ADDR+MOD corresponds to Field 4: compute the final address M, sending M to the last 15 bits of I; load S with M or (M); then inflect (S). Any field except LOCN or TAG may contain octal codes, which will be placed in the corresponding positions of the machine instruction, ignoring any bits which overflow to the left. (An exception to this rule is discussed under ADDR+MOD symbols in Section I.3.) LOCN may also be absolute in certain cases (see the ORG pseudo-order in Section III.4.). If no Field 3 operation is desired, AUX and the comma preceding it may be omitted.

I.2. Symbolic Language

Precise definitions of the allowed symbols are as follows:

Type I: Special symbolic addresses. By convention we recognize the following symbols for "fast" addresses: Z, U, R, S, T4, T5, T6, T7 (A-series); and CC, B1, B2, B3, B4, B5, B6, PF (B-series). These may appear in SETU, ADDR+MOD, and AUX fields. Use of the above symbols of I (in the SETU or AUX fields),

and of the following special register symbols should be restricted to the special significance associated with the Rice Computer:

SL	the sense light register
IL	the indicator light register
ML	the mode light register
TL	the trapping light register
P2	the second pathfinder
X	the increment register
TT	the "to-tape" register
FT	the "from-tape" register

Type II: Special characters. *, a(AP1) or #(AP2), d(AP1), +, -, |, →, (,), "tab", "cr", and , (comma).

Type III: Mnemonic operation codes as listed in Section II; AP1 pseudo-operation codes as discussed in Section III.4.; and symbols previously defined as operation codes by means of a LET (AP2) or EQU(AP1) instruction (see the EQU pseudo-order in Section III.4.).

Type IV: General storage addresses. In AP2, any private name formed by the rules described in the Notes on Genie. In AP1, any upper case Roman letters, which may be followed by upper case Roman letters, or numerals; any number of characters are permissible, but a maximum of six will be retained by AP1. (A letter may not follow a numeral; BA3 is permissible, but not B3A.) Examples: B, M3, COMM, ZETA2. These symbols may appear only in the LOCN or ADDR fields.

I.3. Contents of Instruction.

Each field of the symbolic instruction has a well-defined form, and if this is not recognized by the assembly system, a note is made on the printed listing of the program. The acceptable contents of each field are as follows:

LOCN. May be blank or absolute or symbolic. Absolute LOCN fields are permitted only when an AP1 program is being assembled in absolute form (see the ORG pseudo-order in section III.4.). Symbolic LOCN fields may consist of any Type IV symbol, and may appear in conjunction with a relative numerical part, as "LOOP+1", "EXIT-3". A symbol may not appear in LOCN more than once in any one program unless the correct numerical relationship is specified.

SETU. May be blank, absolute, "I", or F, where F is an A- or B-series symbolic address, or any of the forms -F, |F|, or -|F|. If SETU is blank, the symbol "U" is understood and the octal equivalent 01 is inserted into the machine instruction. I (or |Z|) and -I (or -|Z| have special meanings: I sets U to the numerical integer +1; - I sets U to the numerical integer -1. Note that Z sets U to all zeros; -Z sets U exponent to zero and U mantissa to minus zero, or all ones.

Examples: B1 |T4| -PF -|R| I -I

OPN. May be any absolute octal code or Type III symbol. In the case of conditional transfers, a symbolic operation has the form IF(CCC)TTT where CCC represents test conditions and TTT is a mnemonic for a transfer order. Other symbolic operation codes consist of one or more 3-letter mnemonics. Special symbols such as →, +, -, ",", and +i (where i is an octal integer) are sometimes permitted (see the codes in Section II).

AUX. May be blank, absolute, or one of the forms U→F, R→F, I→Bi, Bi+1, Bi-1, or Bi+X, where Bi stands for one of the Type I B-series symbols, F is any Type I A- or B-series symbol, I refers to the last 15 bits of the instruction register, and X refers to the increment register. Note that R→Z causes R to be cleared to zero.

Examples: U→T4 R→PF I→B1 B2+1 B3-1 B4+X

ADDR+MOD. ADDR may be blank or absolute or symbolic, or the ADDR+MOD field may consist of an octal or decimal number to be used as an operand. MOD is either blank or one or more of the Type I B-series symbols, connected to ADDR by + signs. Special inflections control the IM and IA bits as follows: IM bit 1 is set to 1 whenever the symbol "a" (AP1) or "#" (AP2) appears, or whenever certain OPN mnemonics are used (see the listing of OPN codes in Section II). IM bits 2 (absolute value) and 3 (minus) are controlled by the special forms -Q, |Q|, and -|Q|, where Q is an allowed ADDR+MOD symbol. The IA (indirect addressing) bit is set to 1 whenever the symbol "*" appears in this field.

If ADDR is symbolic, any Type I A-series symbol, any special register symbol, or any Type IV symbol is acceptable. As in LOCN, this field may contain a relative part consisting of an integer preceded by a + or - sign.

If ADDR is absolute, any octal integer of not more than 5 digits, or any decimal integer of absolute value not larger than 32,767, is permissible. Any octal or decimal integer above these limits, or any floating point decimal number (see the DEC pseudo-order in Section III.4.), is treated as an operand; storage space is reserved for it at the end of the program. In this case, the MOD, "a" or "#", and "*" symbols must be omitted, but the other IM inflections may be present.

All characters appearing within parentheses in this field are treated as the Z character, so that an address which is modified by the program may be conveniently noted. For example, (FWA)+B1+B2 is treated as Z+B1+B2. If a symbol appears in ADDR but never in LOCN, a blank location will be reserved at the end of the program; this is true even when such a symbol appears only within parentheses. ADDR and MOD should not both be blank; the Z character may always be used to produce a zero field.

Examples of equivalent AP1 and AP2 ADDR+MOD fields are:

AP1	AP2
COMM+10 or COMM+d8	COMM+8 or COMM++10
-|A+B1-d12| or -|A+B1-14|	-|A+B1-12| or -|A+B1-+14|
a*ZETA	#*ZETA
d48	48
-ad122+B1	-#122+B1
B4+B5	B4+B5
00500	+00500
d2.009027	2.009027
777700000	+777700000
30	24

The only field which may be continued onto another line is "ADDR+MOD, AUX". This is achieved by punching a "cr" followed immediately by three "tab" characters, so that continuation lines will follow under "ADDR+MOD, AUX". Any number of "cr" characters may be punched to help separate code sequences on the printed page; they will appear in symbolic listings, but will be ignored during assembly. Comments punched with a 7th hole in the tape may be included for the guidance of the coder; they are not read by the computer and hence will not appear on machine listings of the program.

TAG. May be blank or symbolic. If blank, the 4th tab punch may be omitted. If symbolic, TAG must be one of the mnemonics TG1, TG2, or TG3. The corresponding tag iwll be placed on the assembled instruction, printed on the octal listing, and punched with the instruction in check-sum format.

II. MNEMONIC OPERATION CODES

The most common Field 2 operations have been assigned symbolic equivalents in AP1 and AP2 for convenience in coding. All Field 2 operations are fully explained in the machine manual. The mnemonics defined in this section are summarized in a chart at the end of the section. These Type III symbols may not be used for any other purpose. Other Field 2 operations may be assigned symbolic equivalents by LET (AP2) or EQU (AP1) statement (see the EQU pseudo-order in Section III.4.); such symbols are then treated as Type III symbols throughout the program in which they have been defined. In the list below, the symbols are followed by their octal equivalents and a brief explanation of their meanings; the indication "a,#" means that the operation symbol automatically causes IM bit 1 to be set to 1, since the operation indicated deals with M rather than with (S).

II.1. Class 0, Tests and Transfers.

The four unconditional transfers are represented by:

		octal codes	
a,#	HTR	00000	Halt and transfer. Halt, setting CC to M when CONTINUE is pressed.
a,#	TRA	01000	Transfer. Set CC to M.
	SKP	02000	Skip. Subtract (S) from (U); then increment CC by 1, skipping the next order.
	JMP	03000	Jump. Subtract (S) from (U); then increment CC by (X), the increment register.

Conditional transfers have the form IF(CCC)TTT where TTT is one of the above transfer mnemonics, and CCC represent one, two, or three test conditions joined by + or X signs. Use of the + sign indicates that the specified transfer is to occur if <u>any</u> of the conditions listed is satisfied; use of the X sign indicates that the specified transfer occurs only when <u>all</u> of the conditions listed are satisfied simultaneously. A single order may not

contain both + and X signs. One condition from each of the first three groups may be specified; or a Group IV mnemonic may be combined with a Group III test as noted. If a TRA or HTR is used, the specified test is made on (U). If a SKP or JMP is used, the specified test is normally performed on (U)-(S). The exceptions to this rule are noted below Group II.

Group I

		octal code	
	PSN	00100	Positive sign. Is the sign bit of U equal to 0?
	MOV	00200	Mantissa overflow. Is Indicator Light #4 on?
	EOV	00300	Exponent overflow. Is Indicator Light #5 on?
	NSN	00500	Negative sign. Is the sign bit of U equal to 1?
	NMO	00600	No mantissa overflow. Is Indicator Light #4 off?
	NEO	00700	No exponent overflow. Is Indicator Light #5 off?

Note that indicator lights are turned off when tested.

Group II

		octal code	
	ZER	00010	Zero. Is (U) mantissa all 1's or all 0's?
	EVN	00020	Even. Is bit 54 of U equal to zero?
a,#	SLN	00030	Sense light on. Are all the sense lights corresponding to 1's in M on?
	NUL	00040	Null. Are all 54 bits of U zero?
	NZE	00050	Non-zero. Is (U) mantissa different from zero?
	ODD	00060	Odd. Is bit 54 of U equal to 1?
a,#	SLF	00070	Sense light off. Are all the sense lights corresponding to 1's in M off?

Note that sense lights are not altered when tested. SLN and

SLF tests are meaningful only with SKP or JMP orders, and in these cases no subtraction takes place. If the NUL test is used with a SKP or JMP order, a logical comparison is made as follows: wherever a bit of R is equal to zero, the bits in corresponding positions of U and S are compared. If (U) is identical with (S) in each of these positions, the resulting (U) is null and the NUL portion of the test is satisfied.

Group III

	octal code	
TG1	00001	Tag 1. Is Indicator Light #1 on?
TG2	00002	Tag 2. Is Indicator Light #2 on?
TG3	00003	Tag 3. Is Indicator Light #3 on?
NTG	00004	No tag. Are Indicator Lights #1, #2, #3 all off?
NT1	00005	No tag 1. Is Indicator Light #1 off?
NT2	00006	No tag 2. Is Indicator Light #2 off?
NT3	00007	No tag 3. Is Indicator Light #3 off?

Note that indicator lights are turned off when tested.

Group IV

	octal code	
POS	00110	Positive or zero. Is (U) mantissa greater than or equal to zero?
NEG	00510	Negative or zero. Is (U) mantissa less than or equal to zero?

A + sign must be used when combining either of these mnemonics with a Group III test.

	octal code	
PNZ	04150	Positive and nonzero. Is (U) mantissa strictly greater than zero?
NNZ	04550	Negative and nonzero. Is (U) mantissa strictly less than zero?

A X sign must be used when combining either of these mnemonics with a Group III test.

II.2. Class 1, Arithmetic.

Any Class 1 mnemonic may be followed by → or +1, to cause storing of the final (U) in the location addressed by M; by +2, storing (U) at location (B6); or by +3, storing (U) at location M+(B6). Octal codes may be joined by a '+' to Class 1 mnemonics for various special operations. If n is such an octal code, the combination appears as

mnemonic +n	in AP1
mnemonic ++n	in AP2

Any floating point mnemonic may be followed by +1j (j=0, 1, 2, or 3), causing (U) to be rounded (before storing); or by +6j, suppressing normalization of the result in U; or by +7j, to obtain rounding without normalization. In addition, SUB may be followed by +400j, and FSB may be followed by +40ij (i=0, 1, 6, or 7; j=0, 1, 2, or 3) to interchange (U) and (S) before subtracting. The Class 1 mnemonics are as follows:

	octal code	
ADD	10000	Add. (U)+(S)→U.
SUB	10100	Subtract. (U)-(S)→U.
BUS	14100	Reverse subtract. (S)-(U)→U.
MPY	10200	Multiply. (U)×(S)→U,R (double length).
IMP	10220	Integer multiply. (U)×(S)→U.
DIV	10300	Divide. Double length (U,R)÷(S)→U, remainder →R.
VID	16300	Reverse divide. (S)÷(U)→U, remainder →R.
IDV	13300	Integer divide. (U)÷(S)→U, remainder →R.
VDI	17300	Reverse integer divide. (S)÷(U)→U, remainder →R.
FAD	10400	Floating add. (U)+(S)→U.
FSB	10500	Floating subtract. (U)-(S)→U.
BSF	14500	Reverse floating subtract. (S)-(U)→U.
FMP	10600	Floating multiply. (U)×(S)→U,R (double length).

	octal code	
FDV	10700	Floating divide. Double length (U,R)÷(S)→U, remainder →R.
VDF	16700	Reverse floating divide. (S)÷(U)→U, remainder →R.

II.3. Class 2, Fetch, Store, Tags.

Any Group I or Group II mnemonic may be followed by a comma and any Group III mnemonic. In addition, any Group I or Group III mnemonic may be followed by → or +1, storing (U) with (ATR) at location M; or by +2, storing (U) with (ATR) at location (B6); or any Group I, II, or III mnemonic may be followed by +3, storing (U) with (ATR) at location M+(B6). Note that all Group I and Group II mnemonics clear (ATR) unless followed by a Group III mnemonic. The Class 2 mnemonics are as follows:

Group I

	octal code	
CLA	21700	Clear and add. Bring (S) to U.
BEU	21000	Bring exponent to U. Exponent portion of (S) replaces exponent portion of (U).
BMU	20700	Bring mantissa to U. Mantissa portion of (S) replaces mantissa portion of (U).
BLU	21400	Bring left half to U. Left half of (S) replaces left half of (U).
BRU	20300	Bring right half to U. Right half of (S) replaces right half of (U).
BIU	20200	Bring inflections to U. Inflection portion of (S) replaces inflection portion of (U).
BAU	20100	Bring address to U. Address portion of (S) replaces address portion of (U).
BNA	21600	Bring non-address to U. All portions of (S) except the address portion replace all portions of (U) except the address portion.

Group I (continued)

	octal code	
BEU, BRU	21300	Bring exponent and right half to U.
BEU, BAU	21100	Bring exponent and address to U.
BLU, BAU	21500	Bring left half and address to U.

Group II

	octal code	
RPE	20701	Replace exponent. Exponent portion of (U) replaces exponent portion of word at location M.
RPM	21001	Replace mantissa. Mantissa portion of (U) replaces mantissa portion of word at location M.
RPL	20301	Replace left half. Left half of (U) replaces left half of word at location M.
RPR	21401	Replace right half. Right half of (U) replaces right half of word at location M.
RPA	21601	Replace address. Address portion of (U) replaces address portion of word at location M.
a,# STO	20001	Store. Store (U) at location M.

Note: "Replace" mnemonics may not be combined with each other.

Group III

	octal code	
ST1	20010	Set Tag 1. Set ATR to 1.
ST2	20020	Set Tag 2. Set ATR to 2.
ST3	20030	Set Tag 3. Set ATR to 3.
WTG	20040	With Tag. Do not change ATR.

Group IV

	octal code	
NOP	30000	No operation. Do not alter (U) or (ATR).
FST	20041	Fetch and store. Bring contents of location M to S; then store (U) with (ATR) at location M.
RWT	21641	Replace address, with tag. Address portion of (U) replaces address portion of word at location M, without changing the tag on the word at location M.

II.4. Class 4, B-Registers, Lights, Special Registers, Shifts.

The Class 4 mnemonics are as follows:

		octal code	
a,#	TSR	40000	Transfer to subroutine. Set PF to (CC); then set CC to M.
a,#	SBi	4000i	Set Bi. Set Bi to M, for i=1, 2, ..., 6.
a,#	SPF	40007	Set PF. Set PF to M.
a,#	ACC	41000	Add to CC. (CC)+M→CC.
a,#	ABi	4100i	Add to Bi. (Bi)+M→Bi, for i=1, 2, ..., 6.
a,#	APF	41007	Add to PF. (PF)+M→PF.
	ERM	00020	Enter repeat mode. Turn on mode light #2.

The ERM mnemonic is meaningful only when joined by a comma to one of the above Class 4 mnemonics.

		octal code	
a,#	SLN	42000	Sense light on. Turn on sense lights corresponding to 1's in M.
a,#	ILN	42001	Indicator light on. Turn on indicator lights corresponding to 1's in M.
a,#	MLN	42002	Mode light on. Turn on mode lights corresponding to 1's in M.
a,#	TLN	42003	Trap light on. Turn on trapping lights corresponding to 1's in M.

		octal code	
a,#	SLF	42004	Sense light off. Turn off sense lights corresponding to 1's in M.
a,#	ILF	42005	Indicator light off. Turn off indicator lights corresponding to 1's in M.
a,#	MLF	42006	Mode light off. Turn off mode lights corresponding to 1's in M.
a,#	TLF	42007	Trap light off. Turn off trapping lights corresponding to 1's in M.

Note that lights corresponding to 0's in M are not affected by the above orders.

		octal code	
a,#	STX	43005	Set X. Set the increment register to M.
a,#	STT	43006	Set TT. Set the to-tape register to M.
a,#	SFT	43007	Set FT. Set the from-tape register to M.

		octal code	
a,#	DMR	44000	Double mantissa right. Arithmetic right shift of (U,R) mantissa M places as diagrammed in the Rice Computer Manual.
a,#	DML	44010	Double mantissa left. Arithmetic left shift of (U,R) mantissa M places as diagrammed in the Rice Computer Manual.
a,#	LUR	45010	Logical U right. Shift (U) right M places, shifting zeros into left end of U.
a,#	LUL	45020	Logical U left. Shift (U) left M places, shifting zeros into right end of U.
a,#	LRR	45001	Logical R right. Shift (R) right M places, shifting zeros into left end of R.
a,#	LRL	45002	Logical R left. Shift (R) left M places, shifting zeros into right end of R.
a,#	LRS	45015	Long right shift. Shift (U,R) right M places, shifting (U) into R and zeros into left end of U.

		octal code	
a,#	LLS	45062	Long left shift. Shift (U,R) left M places, shifting (R) into U and zeros into right end of R.
a,#	CRR	45055	Circle right. Shift (U,R) right M places, shifting (U) into R and right end of (R) into left end of U.
a,#	CRL	45066	Circle left. Shift (U,R) left M places, shifting (R) into U and left end of (U) into right end of R.
a,#	BCT	46000	Bit count. Clear U; shift R right M places; add each 1 which spills from R one at a time into U.

II.5. Class 5, Logic.

Any Class 5 mnemonic may be followed by → or +1, to cause storing of the final (U) at location M; by +2, storing (U) at location (B6); or by +3, storing (U) at location M+(B6). In addition, any Class 5 mnemonic may be preceded by a - sign, causing the final result in U to be complemented (before storing). The Class 5 mnemonics are as follows:

	octal code	
CPL	50100	Complement. Change all 1's in U to 0's and all 0's to 1's.
XUR	54000	Exchange (U) and (R). (U)→R as (R)→U.
LDR	50400	Load R. (S)→R without disturbing (U).
LTi	504i0	Load Ti. (S)→Ti without disturbing (U) or (R), for i=4, 5, 6, 7.
ORU	50010	Or to U. Logical or: for every bit position, a one in U or a one in S (or both) results in a one in that position of U. A zero in any bit position of both U and S results in a zero in that position of U.

	octal code	
AND	50314	And. Logical and: for every bit position, a one in U and a one in S results in a one in that position of U. A zero in any bit position of either U or S results in a zero in that position of U.
XTR	50020	Extract. Wherever a bit of R is equal to one, the bit in that position of S replaces the corresponding bit in U. Other bits of U are unchanged.
SYM SYD	53220 53220	Symmetric difference. For every bit position, a one in U and a zero in S, or vice versa, results in a one in that position of U. Two zeros, or two ones, in corresponding bit positions of U and S result in a zero in that position of U.
SYS	53120	Symmetric sum. For every bit position, a one in U and a zero in S, or vice versa, results in a zero in that position of U. Two zeros, or two ones, in corresponding bit positions of U and S result in a one in that position of U.

II.6. Class 6, Input-Output.

For detailed explanations of reading, printing, punching, plotting, and magnetic tape operation, see the Rice Computer Manual. The Class 6 mnemonics are as follows:

For paper tape,

		octal code	
a,#	RTR	60000	Read triads. Read 1 to 18 triads from paper tape into U.
a,#	RHX	60100	Read hexads. Read 1 to 9 hexads from paper tape into U.
	PHX	60400	Punch hexads. Punch 1 to 9 hexads from (S) onto paper tape.
	PH7	60500	Punch hexads with 7th hole. Punch 1 to 9 hexads, each with a 7th hole, from (S) onto paper tape.
	PTR	60600	Punch triads. Punch 1 to 18 triads from (S) onto paper tape.

Either "Read" mnemonic may be followed by → or +1, storing (U) at location M; by +2, storing (U) at location (B6); or by +3, storing (U) at location M+(B6).

For console typewriter,

		octal code	
	TYP	60700	Type. Type (S) as 18 octal digits on console typewriter.

For printer,

		octal code	
a,#	PRN	61110	Print numeric. Print, using first 32 characters of print wheel, from print matrix beginning at location M; space one line after printing.
a,#	PRA	61210	Print alphanumeric. Print as above, using all characters.
a,#	PRO	61310	Print octal. Print as above, using characters 0-7 only.

		octal code	
	SPA	61010	Space. Advance printer paper one line.
	SP2	61020	Space, format 2. Advance printer paper to next 1/22 page mark.
	SP3	61030	Space, format 3. Advance printer paper to next 1/11 page mark.
	SP4	61040	Space, format 4. Advance printer paper to next 1/6 page mark.
	SP5	61050	Space, format 5. Advance printer paper to next 1/3 page mark.
	SP6	61060	Space, format 6. Advance printer paper to next 1/2 page mark.
	PAG	61070	Page restore. Advance printer paper to next new page.
	DLY	61000	Printer delay. n successive executions of DLY will delay the machine for n-1 tenths of a second.

For magnetic tape,

		octal code	
a,#	WDi	64i00	Write data on MT unit i; i=1, 2, 3.
	WMi	64i20	Write marker from last 8 bits of (S) on MT unit i; i=1, 2, 3.
a,#	RDi	65i00	Read data from MT unit i; i=1, 2, 3.
	SMi	66i00	Search for marker in last 8 bits of (S) on MT unit i; i=1, 2, 3.
	RWi	66i01	Rewind tape on MT unit i; i=1, 2, 3.
	BCK	60040	Backward. Perform operation in backward direction.
	NST	65004	No store. Do not store to memory. This is meaningful only for read MT orders.

For oscilloscope plot,

	octal code	
PLT	67000	Plot on oscilloscope.
ADV	67700	Advance movie film.

II.7. Class 7, Analog Input.

Any Class 7 mnemonic may be followed by → or +1, to cause storing of the final (U) at location M; by +2, storing (U) at location (B6); or by +3, storing (U) at M+(B6). This class deals with various instructions used in conjunction with operation of the analog-to-digital converter. The Class 7 mnemonics are as follows:

	octal code	
WAT	71000	Wait. Machine will wait until the next pulse from a crystal-controlled 1 kc. pulse generator before exiting Field 2.
LS1	72010	Special fast arithmetic left shifts of the double-length (U,R). Shifts are 8 bits at a time. LSi indicates i shifts of 8 bits. These shifts are principally used in unpacking converted data. The mnemonics may be combined to get different length shifts: LS4, LS1 would give 5 left shifts of 8 bits (total: 40 bits). These shifts do not pass through the exponents of U or R nor through the sign of R, but do shift into the sign of U.
LS2	72020	
LS4	72040	
MCN	72110	Manual conversion. An A-to-D conversion of the channel specified by (S) will be performed.
ACN	72364	Automatic conversion. Six conversions from channels 1 through 6 will be performed.

Conversion results will be packed into U as follows: The 8 bits (sign plus 7 bits) resulting from each conversion will be packed into the mantissa with the bits resulting from the first conversion farthest to the left and the bits resulting from last conversion in the right-most 8 bits of U. The U exponent will be set to 77. The R mantissa is used.

There are sixteen channels into the converter. The channel to be converted is specified by the right-most 16 bits of S. Channel 1 corresponds to S_{m47}, Channel 2 to S_{m46}, etc.

In addition to the normal store options, operations may be performed with the 72xxx orders as follows:

72xxx + 400	(S) will be sent to U before performing any other operation.
72xxx + 200	(S) will be cleared and a 1 sent to S_{m47}.
72xxx + 4	(S) will be logically shifted 1 to the left each time (U,R) is shifted 8 to the left. Notice that this feature can be used to sample consecutively numbered channels automatically.

II.8. Summary of Operation Codes.

The accompanying chart summarizes the Field 2 mnemonics available in AP1 and AP2. If an operation code is followed by the symbol "@", the corresponding mnemonic generates a "a" bit; that is, causes IM bit 1 to be set to 1.

The symbol "→" following an operation mnemonic of class 1, 2, 5, 6 causes a final store of U to M.

The symbol "-" preceding a class 5 operation mnemonic causes a final logical complement of U.

Operation mnemonics will be compounded by a logical OR of their equivalents. Mnemonics whose equivalents have the symbol "-" instead of certain digits indicate that these mnemonics must be compounded with those they are grouped with to be meaningful. Most mnemonics are compounded with commas. In class 0, the test may be compounded with "+" or "," to give an "ANY" test, or with "X" to give an "ALL" test. The mnemonics "POS" and "NEG" are compound "ANY" tests and the mnemonics "PNZ" and "NNZ" are compound "ALL" tests.

SUMMARY OF OPERATION CODES

CLASS 0		CLASS 0	CLASS 1	CLASS 2	CLASS 4	CLASS 4	CLASS 5	CLASS 6
HTR	00000@	PSN 0-1--	ADD 10000	STO 20001@	TSR 40000@	DMR 44000@	LDR 50400	RTR 60000@
TRA	01000@	MOV 0-2--	SUB 10100	FST 20041	SBi 4000i@	DML 44010@	LT4 50440	RHX 60100@
SKP	02000	EOV 0-3--	MPY 10200		SPF 40007@	LUR 45010@	LT5 50450	PHX 60400
JMP	03000		DIV 10300	BAU 20100	ACC 41000@	LUL 45020@	LT6 50460	PH7 60500
IF(ANY)HTR	00---@	NSN 0-5--	BUS 14100	BIU 20200	ABi 4100i@	LRR 45001@	LT7 50470	PTR 60600
IF(ANY)TRA	01---@	NMO 0-6--	IMP 10220	BRU 20300	APF 41007@	LRL 45002@		TYP 60700
IF(ANY)SKP	02---	NEO 0-7--	IDV 13300	BMU 20700	ERM 4--2-	LRS 45015@	ORU 50010	
IF(ANY)JMP	03---		VID 16300	BEU 21000		LLS 45062@	AND 50314	PRN 61110@
IF(ALL)HTR	04---@	ZER 0--1-	VDI 17300	BLU 21400	SLN 42000@	CRR 45055@	SYM 53220	PRA 61210@
IF(ALL)TRA	05---	EVN 0--2-		BNA 21600	ILN 42001@	CRL 45066@	SYD 53220	PRO 61310@
IF(ALL)SKP	06---	SLN 0--3-@	FAD 10400	CLA 21700	MLN 42002@	BCT 46000@	SYS 53120	SPA 61010
IF(ALL)JMP	07---	NUL 0--4-	FSB 10500		TLN 42003@		XTR 50020	SP2 61020
		NZE 0--5-	FMP 10600	RPL 20301	SLF 42004@	CLASS 7	CPL 50100	SP3 61030
POS	0-11-	ODD 0--6-	FDV 10700	RPE 20701	ILF 42005@	WAT 71000	XUR 54000	SP4 61040
NEG	0-51-	SLF 0--7-@	BSF 14500	RPM 21001	MLF 42006@	ACN 72364		SP5 61050
			VDF 16700	RPR 21401	TLF 42007@	MCN 72110	CLASS 6	SP6 61060
PNZ	0-15-	TG1 0---1		RPA 21601		LS1 72010	WDi 64i00@	PAG 61070
NNZ	0-55-	TG2 0---2		RWT 21641	STX 43005@	LS2 72020	WMi 64i20	
		TG3 0---3			STT 43006@	LS4 72040	RDi 65i00@	PLT 67000
		NTG 0---4		STi 2--i-	SFT 43007@		NST 65-04	ADV 67700
		NT1 0---5		WTG 2--4-			SMi 66i00	
		NT2 0---6		NOP 20040			RWi 66i01	
		NT3 0---7					BCK 6--4-	

The tables on this page summarize the options available in SETU (Field 1), AUXILIARY (Field 3), and SETS (Field 4). In the tables

A indicates the full length special registers Z,U,R,S,T4,T5,T6,T7 specified in the second triad by 0,1,2,3,4,5,6,7.

B and Bi indicate the short index registers CC,B1,B2,B3,B4,B5,B6, PF specified in the second triad by 0,1,2,3,4,5,6,7.

M indicates the number formed in the address field of the instruction. (M) indicates the contents of the memory location numbered M.

Exceptions are R→Z, 10 in field 3 and |Z|, 20 and -|Z|, 30 in field 1. R→Z has the result that R is cleared to Z. |Z| has the result that a fixed point integer 1 goes to U. -|Z| has the result that a fixed point integer -1 goes to U.

1st Triad (SETU)		Field 1	
A	0	B	4
-A	1	-B	5
|A|	3	|B|	6
-|A|	4	-|B|	7

1st Triad (AUXILIARY)		Field 3	
U→A	0	U→Bi	4
R→A	1	R→Bi	5
Bi+1	2	Bi-1	6
Bi+X	3	I→Bi	7

1st Triad (SETS)		Field 4	
(M)	0	M	4
-(M)	1	-M	5
|(M)|	2	|M|	6
-|(M)|	3	-|M|	7

III. PLACER AND ASSEMBLY OPERATIONS

III.1. Genie Placer.

The typing of Genie tapes containing AP2 instructions, and the handling of such tapes by the Genie PLACER system and the Genie compiler, are just as described in the Notes on Genie. AP2 instructions may appear anywhere between the 'SEQ' and 'END' of a Genie program and may be interspersed with Genie language commands. One word of caution: though each AP2 command produces only one machine instruction, a single Genie command may produce several machine instructions. Therefore relative addressing and skip and jump commands should be used only within a sequence of AP2 instructions.

Genie PLACER contains the Back-Translator, which provides for conversion of absolute machine language to AP1. This is explained in the Notes on Genie.

III.2. AP1 PLACER.

The AP1 PLACER system is located on the MT system magnetic tape at block 100.02. There are three major differences between Genie PLACER and AP1 PLACER: to AP1 PLACER, SL6 means "Assemble" rather than "Compile"; AP1 PLACER has no SL7 option; and the AP1 PLACER lister prints only valid AP1 characters, replacing any other characters with the symbol ←.

When AP1 PLACER is read into memory, program *240 is executed, and the stop

```
(I):    00    HTR    CC
```

occurs. The set of options to be exercised should then be designated in the sense lights:

SL1	Read symbolic tape.
SL2	Edit.
SL3	Punch (edited) symbolic tape.
SL4	List (edited) symbolic tape.
SL5	Check (edited) symbolic tape punched.
SL6	Assemble (edited) symbolic tape.

The original tape to be processed should be placed in the reader. Pushing CONTINUE causes the specified operations to be carried out in order as described below:

SL1, READ. The tape to be read must contain only one symbolic program, this begun with one carriage return and terminated by two carriage returns. All information beyond the last CR is ignored by the system. When the reading is complete, the system has in the machine a tape image.

SL2, EDIT. The stop

(I): 02 HTR CC

occurs. The edit tape is placed in the reader. Pushing CONTINUE causes this tape, which must contain only the corrections for the tape image in the machines to be read. When reading is complete, PLACER's tape image in the machine is edited.

Each correction is specified by three octal parameters: the initial carriage return number (i), the final carriage return number (f), and the number of lines in the symbolic correction (n). A line in a symbolic tape is terminated by a carriage return, these being numbered from 1 on listings. The n lines of a correction will replace the portion of the program read from and not including carriage return i through carriage return f. Note that n=0 effects a deletion. The last line of a symbolic tape must not be replaced. On a single edit tape, the f of one correction may not equal the i of another correction. The format for punching the correction parameters is:

(l.c.) i (sp) f (sp) n (cr)

SL3, PUNCH. The tape image in the machine is punched out on paper tape.

SL4, LIST. The tape image in the machine is listed on the fast line printer with carriage return numbers.

SL5, CHECK. In a series of operations, CHECK is initially bypassed if the tape to be checked is not in the reader. If all other operations are complete and the tape is still not in the

reader, the halt

(I): 05 HTR CC

occurs. The tape that is read is compared to the tape image in the machine, and an error print is given if the comparison fails.

SL6, ASSEMBLY. The tape image in the machine is assembled. If only one sense light option is requested, the stop

(I): 0i HTR CC

(for SLi on) occurs; other sense lights may then be set for special forms of output. Pushing CONTINUE then causes the operation indicated to be carried out. The options are:

- For LIST (SL4 on), setting SL15 on when the stop

  (I): 04 HTR CC

  occurs causes double spacing on the listing.
- For ASSEMBLE (SL6 on), the following operation may be exercised by turning on the appropriate sense lights (in addition to lights 14 and 15 which are turned on automatically) when the stop

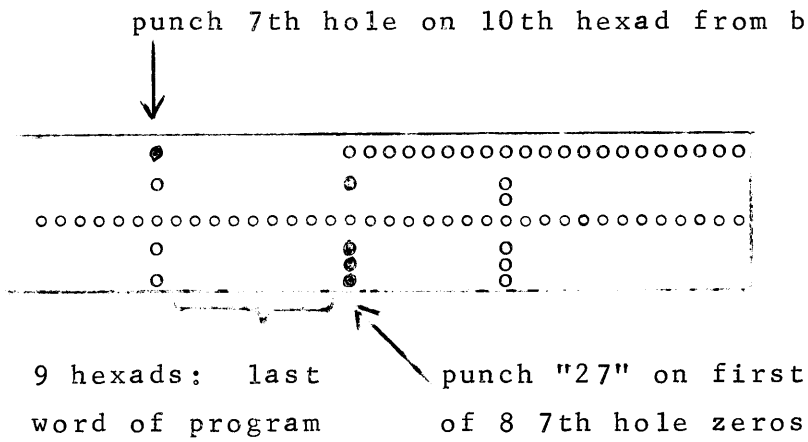
  (I): 06 HTR CC

  occurs:

SL9 on: Print with double (instead of single) spacing.

SL11 on: Do not punch assembled program.

SL13 on: Punch self-loading tape (permissible only for absolute programs; see the ORG pseudo-order below). The paper tape produced by AP1 is punched in hexad form with check-sum, and is normally preceded by a SPIREL control word. By assembling an absolute program with SL13 on, a tape will be produced which will load independently of SPIREL (by using the LOAD switch) after the following changes are made with the hand punch:



III.3. AP1 Assembly Output.

The printed output from AP1 assembly is interpreted as follows:

Error indications. An error indication is produced by apparent errors in syntax or sequencing. The type of error is briefly indicated, followed by a line number from which it should be possible to detect the source of the error.

Note that line numbers refer to the partially assembled program which still contains pseudo-orders; location numbers refer to the final form of the program containing only valid machine instructions.

Symbol table. The table of symbols is printed out in seven columns giving information relevant to the symbols defined in the program:

(a) The relative position in the table.

(b) The symbol.

(c) A number (usually 0) which determines the type of object for which the symbol stands.

(d) The equivalent assigned to the symbol (5 octal digits).

(e) A number (usually 0) which determines whether or not an equivalent has been assigned. A number 2 indicates that a symbol remains unassigned and is a possible error in the final program.

(f) An 18 digit octal number. The first 5 digits indicate the line at which an equivalent was assigned.

(g) A number which indicates how (if at all) the equivalent was assigned:
0: by appearing in the LOCN field of an order.
1: by appearing in the LOCN field of an EQU pseudo-order in which the address was symbolic (see section on pseudo-orders).
2: by appearing in the LOCN field of an EQU pseudo-order in which the address was numeric (see section on pseudo-orders).

Assembled program listing. Five columns are printed, giving:
(a) The line count in octal.
(b) The symbolic location (if any exists).
(c) The location count in octal.
(d) The instruction in octal, broken into fields, with tag.
(e) The symbolic address (if any exists).

III.4. AP1 Pseudo-orders.

Differences between line and location numbers, and skips in the sequence of line numbers, are caused by pseudo-orders as described below. These special instructions govern the process of AP1 assembly and facilitate the handling of blocks of various types of data within AP1 programs.

ORG and END. All programs to be assembled by AP1 must be preceded by an ORG (Origin) order and terminated by an END order. In each case the remaining fields in the symbolic instruction are interpreted in a special way. Each of these orders advances the line count by 1; the location count is not affected.

The function of ORG is (a) to initialize the assembly process, (b) to identify the program which follows, (c) to determine whether it is to be assembled in relative or absolute final form, and (d) to give an approximate indication of its maximum size. The ORG order is preceded by a "cr" and an "uc" or "lc" punch (upper or

lower case); the order itself has the form

1	s	ORG	m,n
|cr	|1st tab	|2nd tab	|3rd tab

where 1, s, m, and n stand for either absolute or blank fields (symbolic fields are not allowed). If 1 (LOCN) is not zero, it is taken to be a SPIREL codeword address for the relativized routine which follows; in this case m must be zero. The SETU field, s, is always blank or zero. If m (ADDR) is not zero, the following code is assembled in absolute form starting in location m; in this case 1 must be zero. (In a relativized program, if an order in location P refers in Field 4 to location Q, it is through a Control Counter reference of the form CC+(Q-P)-1. In an absolute program, the Field 4 reference is directly to location Q; and absolute LOCN fields, giving the location of an instruction in 5-digit octal form, are permitted.) If n is blank, it is assumed that the total number of lines in the following program is octal 200 or less (d 128). If n is not blank, its value is used in taking storage space for the program which follows. The value of n may be greater or less than 200, but not greater than 10,000 (d 4096).

The END order has the form

b	b	END	cr	cr
|cr	|1st tab	|2nd tab		

where b stands for a blank field. "END" must be immediately followed by two (or more) carriage returns.

EQU. The Equivalance order gives a numeric equivalent for a symbol or equates one symbol to another. The order has the form

1	b	EQU	m
|cr	|1st tab	|2nd tab	|3rd tab

where 1 (LOCN) is the symbol defined by the pseudo-order, b (SETU) is blank, and m (ADDR) is either absolute or a symbol whose value has previously been assigned, through its appearance in the LOCN field of another order. Then 1 is immediately assigned the value m. If m is a 5-digit octal OPN code, then the symbol 1 may appear in Field 2 of any order following the EQU order and will be replaced during assembly by the octal code for which it stands.

This order advances the line count by 1; the location count is not affected.

BSS and BES. Either of these orders inserts a block of zero words into the body of the program. BSS (Block started by symbol) and BES (Block ended by symbol) have the form

1	b	XXX	m
cr	1st tab	2nd tab	3rd tab

where 1 (LOCN) is blank or symbolic, b (SETU) is blank, and m (ADDR) is either absolute or a previously assigned symbol. The value of m is the number of zero words to be inserted; if 1 is symbolic, it is assigned as if the LOCN field had been associated with the first (BSS) or last (BES) word in the block. Each of these orders advances the line count by m+1, and the location count by m.

BCD, FLX, REM. These orders deal with alphanumeric data and have the form

1	b	XXX	m
cr	1st tab	2nd tab	3rd tab

where b (SETU) is always blank; in each case, the mnemonic must be followed by a "tab" character, and after that all characters (in the ADDR field m) are read and stored, 9 characters per word. Any occurrence of the "cr tab tab tab" sequence is replaced by a "space", and the string of characters is terminated by a true carriage return, allowing more than one line of data to be given. For BCD (Binary Coded Decimal), each character is converted to a corresponding printer hexad; if 1 (LOCN) is symbolic, it is assigned as if associated with the first word stored. For FLX (Flexowriter), all codes (including case shifts, etc.) are preserved without conversion; 1 may be symbolic as for BCD. For REM (Remarks), 1 must be blank; this order is used only to obtain printed comments in the program listing, and does not introduce any data into the final program. These orders advance the line

counter by n+1, where n is the number of 9-character words stored; in addition, BCD and FLX advance the location counter by n.

DEC and OCT. The Decimal and Octal orders are used for inserting numerical data into the body of the program. They have the form

1	b	XXX	m
cr	1st tab	2nd tab	3rd tab

where 1 (LOCN) is blank or symbolic, b (SETU) is blank, and m (ADDR) consists of a list of one or more octal or decimal numbers. If 1 is symbolic, it is assigned as if associated with the first number in the list. Each number must be separated from its successor by a comma, and each will occupy a separate word in the final program. Continuation lines should not be used; for long lists of numbers, several DEC or OCT pseudo-orders in succession may be used to produce a continuous block of data. An octal number consists of one to 18 octal digits. A decimal integer consists of one to 14 decimal digits; a floating point decimal number, of one to 14 significant figures and a decimal point. If the list m consists of n numbers, either of these orders will advance the line count by n+1 and the location count by n.

IV. AP1 AND AP2 CODING EXAMPLES

The first example below is a typical AP1 program; the second shows the use of AP2 instructions within a roughly equivalent Genie program. For both examples, the printer listing of the symbolic tape and the assembly or compilation output are reproduced.

3/24/64 13.54

```
                                                                    1
230                 ORG                                             2
                    REM             T4 EQUALS THE SUM OF T5         3
                                    TO THE POWER I FOR I FROM       4
                    REM             ZERO TO J WHERE SL 1, 2,        5
                                    OR 3 ON INDICATES J             6
POLY                IF(SLN)SKP      40000                           7
                    TRA             TWO                            10
ONE         T5      FAD             d1,0,U+T4                      11
                    SLF             40000                          12
                    TRA             EXIT                           13
TWO                 IF(SLN)SKP      20000                          14
                    TRA             THREE                          15
            T5      FAD             d1,0                           16
                    FMP             T5                             17
                    FAD             d1,0,U+T4                      20
                    SLF             20000                          21
                    TRA             EXIT                           22
THREE               IF(SLN)SKP      10000                          23
                    TRA             EXIT                           24
            T5      FAD             d1,0                           25
                    FMP             T5                             26
                    FAD             d1,0                           27
                    FMP             T5                             30
                    FAD             d1,0,U+T4                      31
                    SLF             10000                          32
EXIT                TRA             PF                             33
                    END                                            34
                                                                   35
```

α

PROGRAM 230 3/24/64 13.55

```
 1301        POLY      0     1     0          17000000000000     0
 1302        TWO       0     6     0          24000000000000     0
 1303        ONE       0     3     0          21000000000000     0
 1304              C   0    26     0          45000000000000     0
 1305        EXIT      0    25     0          43000000000000     0
 1306        THREE     0    15     0          33000000000000     0
T4 EQUALS THE SUM OF T5 TO THE POWER I FOR I FROM
ZERO TO J WHERE SL 1, 2,OR 3 ON INDICATES J
          17   POLY      1      1020300040004000
          20             2      1010000040010000 3      TWO
          21   ONE       3      5104000400010002 2           0
          22             4      1420040040004000
          23             5      1010000040010001 7      EXIT
          24   TWO       6      1020300040002000
          25             7      1010000040010000 5      THREE
          26            10      5104000000100015           0
          27            11      1106000000000005
          30            12      1104000400010001 3           0
          31            13      1420040040002000
          32            14      1010000040010001 0      EXIT
          33   THREE     15      1020300040001000
          34            16      1010000040010000 6      EXIT
          35            17      5104000000100006           0
          36            20      1106000000000005
          37            21      1104000000100004           0
          40            22      1106000000000005
          41            23      1104000400010000 2           0
          42            24      1420040040001000
          43   EXIT      25      1010000042000000
          45             26     1001000000000000
```

```
                                                    03/24/64 13,55
                                                                    1
          DEFINE                                                    2
POLY(P,Q),=SEQ                                                      3
          REM    P=SUM Q^I FOR I=0,,,,J WHERE SL 1, 2, OR 3         4
                                ON INDICATES J                      5
          CC=*ONE ,I,F SL^1, *TWO ,I,F SL^2, *THREE ,I,F SL^3, *END
ONE       P=Q+1,0                                                   7
                SLF             +40000                             10
          CC=*END                                                   11
TWO       P=Q^2+Q+1,0                                               12
                SLF             +20000                             13
THREE     P=Q^3+Q^2+Q+1,0                                           14
                SLF             +10000                             15
END                                                                 16
          DEFINE                                                    17
LEAVE                                                               20
                                                                    21
```

```
POLY     START NEW PROGRAM,                                  3/24/64
*BGIN    PROGRAM SEQUENCE,
ONE      PROGRAM SEQUENCE,
TWO      PROGRAM SEQUENCE,
THREE    PROGRAM SEQUENCE,
END      PROGRAM SEQUENCE,


POLY  ,=  0    *BGIN     1     10 01000 02 4400 00136
          1              2     01 40007 00 4100 77764
          2              3     47 21641 00 0001 00054     END    +     1
P=SUM(Q*I), FOR I=0*J WHERE SL 1, 2, OR 3 ON INDICATES J
          3              4     01 21700 00 0000 77770
         14              5     01 45015 00 4000 00016
         15              6     01 50010 00 4000 77776
         16              7     01 01060 00 4001 00001
         17             10     01 01000 00 4001 00002
         20             11     01 21700 00 4001 00021     ONE
         21             12     01 01000 00 4001 00017
         22             13     01 21700 00 0000 77770
         23             14     01 45015 00 4000 00015
         24             15     01 50010 00 4000 77776
         25             16     01 01060 00 4001 00001
         26             17     01 01000 00 4001 00002
         27             20     01 21700 00 4001 00017     TWO
         30             21     01 01000 00 4001 00010
         31             22     01 21700 00 0000 77770
         32             23     01 45015 00 4000 00014
         33             24     01 50010 00 4000 77776
         34             25     01 01060 00 4001 00001
         35             26     01 01000 00 4001 00002
         36             27     01 21700 00 4001 00016     THREE
         37             30     01 01000 00 4001 00001
         40             31     01 21700 00 4001 00025     END
         41             32     01 20040 40 0000 00000
          0    ONE      33     01 21740 00 0001 00026     *ONEF
          1              34     01 10400 00 0600 00001     Q
          2              35     01 30001 00 4600 00000     P
          3              36     01 42004 00 4000 40000
          4              37     01 21700 40 4001 00017     END
          0    TWO      40     01 21700 06 0600 00001     Q
          1              41     06 10600 00 0000 00006     T6
          2              42     01 10400 00 0001 00017     *ONEF
          3              43     01 10400 00 0000 00006     T6
          4              44     01 30001 00 4600 00000     P
          5              45     01 42004 00 4000 20000
          0    THREE    46     01 21700 06 0600 00001     Q
          1              47     06 40021 07 4000 00002
          2              50     01 10600 61 0000 00007
          3              51     01 10400 04 0001 00010     *ONEF
          4              52     06 10600 00 0000 00006     T6
          5              53     01 10400 00 0000 00004     T4
          6              54     01 10400 00 0000 00006     T6
          7              55     01 30001 00 4600 00000     P
         10              56     01 42004 00 4000 10000
          0    END      57     01 01000 00 4400 00137
          1              60     01 40006 00 4000 00000                Y
          2              61     07 01000 00 4200 00000
```

```
 POLY     SYMBOL TABLE.
104         P          2      0     0                          0     0
105         Q        102      1     0                          0     0
106         .BGIN    100      1     3                          0     0
107         ONE      100     33     3                          0     0
110         TWO      100     40     3                          0     0
111         THREE    100     46     3                          0     0

END OF DEFINITION SET.          EXTERNAL SYMBOLS.
  103         POLY     10      2     0                          0     0
```

Δ

TRACE PROGRAM July, 1964

SUPPLEMENTAL SPIREL NOTE No. 1

Option for output of index register contents in tracing.
Add to section V.2., Tracing in USE OF SPIREL.

* * * * * * * * * * * * *

If ML13 is off while tracing, the standard eight-field trace output will be printed. If ML13 is on while tracing, the fields (S), (V'), and (R') will be replaced by (B1'), (B2'), (B3'), (B4'), (B5'), (B6'), (PF') -- the contents of the seven B-registers after execution of the instruction.

In all trace output, the instruction field (I) is formatted into fields; the address of the "next instruction" appears only for non-sequential transfers; and the tag at location M, if not zero, is printed immediately after address M.

MNEMONIC OPERATION CODES July, 1964

SUPPLEMENTAL ASSEMBLY LANGUAGE NOTE No.1

Clarification of octal codes for arithmetic inflections in Class 1. Replace discussion of specific inflections before list of octal codes in section II.2. Class 1, Arithmetic in MNEMONIC OPERATION CODES.

* * * * * * * * * * * * * *

Any Class 1 mnemonic may be followed by +1j (j=0,1,2, or 3) causing the last bit of U to be set to 1 (rounded) after the operation but before storing. After floating point mnemonics +4j suppresses normalization of the result, +5j rounds and suppresses normalization. Class 1 contains several esoteric features not covered here. Consult the Computer Manual for details.