HP-71 ASSEMBLER INSTRUCTION SET — CHAPTER 4

This chapter describes the HP-71 assembler instruction set. The instruction mnemonics shown are those provided by the assembler used by the HP-71 software development team (which is available by special arrangement with HP). Almost all the mnemonics shown are also supported by the HP-71 FORTH/Assembler ROM.

4.1 Instruction Syntax

4.1.1 Labels and Symbols

A label is a symbolic name for a numeric value. A label acquires its value by appearing in the label field of certain statements. The word "symbol" is a general term for a label, and the two are used interchangeably.

Labels are one to six alphanumberic characters with the following restrictions: the characters comma (,), space ( ) and right parenthesis are prohibited and the first character cannot be equal sign (=), sharp (#), single quote ('), left parenthesis, or the digits 0 through 9.

A label may be immediately preceded by an equal sign which declares the label to be an external symbol. An external symbol defined in one module may be referenced as an external symbol by another module. Such references are resolved when the modules are linked together. Certain HP-71 assemblers, such as the FORTH/ASSEMBLER ROM, have no associated linker and therefore do not support external symbols. In this case, any leading equal sign is ignored.

When a label is used as part of an expression, parentheses are required to delineate it. That is, AD1-10 is a label but (AD1)-10 is a computed expression.

4.1.2 Comments

A comment line begins with an asterisk (*) in column one, and may occur anywhere. An in-line comment may begin with any non-blank character and must follow the modifier field of an instruction (or the opcode if no modifier is required).

4.1.3 Expressions

Wherever an expression may appear in the modifier field of an instruction, it is represented by the symbol "expr" in the instruction descriptions below. Expressions consist of:

EXPRESSION COMPONENTS

Item	Examples
decimal constants	23434
hexadecimal constants	#1FF0 (less than #100000)
ascii constants	\AB\ (3 or less characters) 'AB' (3 or less characters)
operators	+ addition - subtraction % *256+ * multiplication / integer division ^ exponentiate & and ! or
*	Current assembly program counter
label	Symbol defined in the label field of an instruction
(expression)	Parenthesized expression

Two classes of instructions require a modifier field which contains a constant of a specific type that does not conform to the above rules. These are:

a) String constant which can exceed 3 characters

```
LCASC    'ascii'      or
LCASC    \ascii\

NIBASC   'ascii'      or
NIBASC   \ascii\
```

b) Unconditional Hex constant

```
LCHEX    4FFFFF
NIBHEX   4FFFFF
```

4.1.4 Sample Line Image

The format below is the recommended column alignment; however, the assembler is "free format" and only a space is required to delimit the different fields. A label, if present, must start by column 2.

```
1      8      15           31                              80
v      v      v            v                               v
-----------------------------------------------------------------
label  opcode  modifier     comments
-----------------------------------------------------------------
```

4.2 Explanation of Symbols

In the following descriptions of the HP-71 assembler mnemonics, these symbols have the following meanings unless specified otherwise. In particular, note the symbols used to indicate the various values encoded within the assembled opcodes.

a — The hex digit used to encode the field selection in the assembled opcode of an instruction. See the Field Select Table in the next section for details.

b — The hex digit used to encode the field selection in the assembled opcode of an instruction. See the Field Select Table in the next section for details.

d — The number of digits represented by a field selection field. Used in calculating the execution cycle time

of some instructions. See the Field Select Table in the next section for details. When used in an extended field selection fsd, represents an expression which indicates the number of nibbles of the register that will be affected by the instruction, proceeding from the low-order nibble to higher-order nibbles.

expr — An expression that evaluates to an absolute or relocatable value, usually less than or equal to 5 nibbles in length.

fs — Field selection symbol. See the Field Select Table in the next section for details.

fsd — Extended field selection symbol. Represents either a normal field selection symbol fs, or an expression that gives the number of digits d of the register that will be affected by the instruction, proceeding from the low-order nibble to higher-order nibbles.

hh — Two-digit hex constant, such as 08 or F2. Within an opcode represents the hex digits used to store the value of the expression in the opcode in reverse order (see "Loading Data From Memory").

hhhh — Four-digit hex constant, such as 38FE. Within an opcode, represents the hex digits used to store the value of the expression in the opcode in reverse order (see "Loading Data From Memory").

hhhhh — Five-digit hex constant, such as 308FE. Within an opcode, represents the hex digits used to store the value of the expression in the opcode in reverse order (see "Loading Data From Memory").

label — A symbol defined in the label field of an instruction.

m — A one-digit decimal integer constant.

n — Represents an expression that evaluates to a 1-nibble value, unless specified otherwise. Within an opcode, represents the hex digit used to store the assembled value of the expression in the opcode.

nn — Represents an expression that evaluates to a 2-nibble value, unless specified otherwise. Within an opcode, represents the hex digits used to store the assembled value of the expression in the opcode.

nnnn — Represents an expression that evaluates to a 4-nibble value, unless specified otherwise. Within an opcode,

represents the hex digits used to store the assembled value of the expression in the opcode.

nnnnn Represents an expression that evaluates to a 5-nibble value, unless specified otherwise. Within an opcode, represents the hex digits used to store the assembled value of the expression in the opcode.

4.2.1 Field Select Table

The following symbols are used in the instruction descriptions to denote the various possible field selections.

There are two ways in which field selection is encoded in the opcode of an instruction. These two patterns are shown in the table below, and are designated by the letter 'a' or 'b' in the opcode value given in the mnemonic descriptions below.

FIELD SELECT TABLE

Field	Name and Description	Opcode Representation (a)	Opcode Representation (b)	Number of Digits (d)
P	Pointer Field. Digit specified by P pointer register.	0	8	1
WP	Word-through-Pointer Field. Digits 0 through (P).	1	9	(P+1)
XS	Exponent Sign Field. Digit 2.	2	A	1
X	Exponent Field. Digits 0 - 2.	3	B	3
S	Sign Field. Digit 15.	4	C	1
M	Mantissa Field. Digits 3 - 14.	5	D	12
B	Byte Field. Digits 0 - 1.	6	E	2
W	Word Field. All digits.	7	F	16

The above field selects generally share the same opcode, with one

nibble of the opcode containing the 'a' or 'b' value as specified in the table. The 'A' field select, however, generally is specified by a different opcode altogether. This has the effect of shortening and speeding up execution of 'A' field select manipulations, optimizing the computer for address and 5-nibble calculations.

4.3 Instruction Set Overview

The following pages briefly summarize the HP-71 instruction set. For further details please refer to the Mnemonic Dictionary which follows this summary.

4.3.1 GOTO Instructions

```
----
 1   = Statement Label
----
```

GOTO	label	Short unconditional branch
GOC	label	Short branch if Carry set
GONC	label	Short branch if no Carry set
GOLONG	label	Long GOTO
GOVLNG	label	Very long GOTO
GOYES	label	Short branch if test true (must follow a Test Instruction)

4.3.2 GOSUB Instructions

GOSUB	label	Short transfer to subroutine
GOSUBL	label	Long GOSUB
GOSBVL	label	Very long GOSUB

4.3.3 Subroutine Returns

RTN	Unconditional return
RTNSC	Return and set Carry
RTNCC	Return and clear Carry
RTNSXM	Return and set XM bit (Module Missing)
RTI	Return and enable interrupts
RTNC	Return if Carry set

```
RTNNC           Return if no Carry set
RTNYES          Return if test true (must follow a
                  Test Instruction)
```

4.3.4 Test Instructions

All test instructions must be followed with a GOYES or a RTNYES instruction. Although they appear to be two statements, in fact they combine to be one. Each test adjusts the Carry bit when performed: Carry is set if the test is true, and cleared if false.

4.3.4.1 Register Tests

```
----
 r,s = A,B,C  or  (r,s) = (C,D),(D,C)
 fs  = Field Select
----

?r=s   fs       Equal
?r#s   fs       Not equal
?r=0   fs       Equal to zero
?r#0   fs       Not equal to zero
?r>s   fs       Greater than
?r<s   fs       Less than
?r>=s  fs       Greater than or equal
?r<=s  fs       Less than or equal
```

4.3.4.2 P Pointer Tests

```
----
 0 <= n <= 15
----

?P=    n        Is P Pointer equal to n?
?P#    n        P Pointer not equal to n?
```

4.3.4.3 Hardware Status Bit Tests

```
?XM=0           Module Missing bit equal to zero?
?SB=0           Sticky Bit equal to zero?
?SR=0           Service Request bit equal to zero?
?MP=0           Module Pulled bit equal to zero?
```

4.3.4.4 Program Status Bit Tests

```
----
 0 <= n <= 15
----
```

?ST=1	n	Status n equal to 1?
?ST=0	n	Status n equal to 0?
?ST#1	n	Status not equal to 1?
?ST#0	n	Status not equal to 0?

4.3.5 P Pointer Instructions

```
----
 0 <= n <= 15
----
```

P=	n	Set P Pointer to n
P=P+1		Increment P Pointer, adjust Carry
P=P-1		Decrement P Pointer, adjust Carry
C+P+1		Add P Pointer plus one to A-field of C
CPEX	n	Exchange P Pointer with nibble n of C
P=C	n	Copy nibble n of C into P Pointer
C=P	n	Copy P Pointer into nibble n of C

4.3.6 Status Instructions

4.3.6.1 Program Status

```
----
 0 <= n <= 15
----
```

ST=1	n	Set Status n to 1
ST=0	n	Set Status n to 0
CSTEX		Exchange X field of C with Status 0-11
C=ST		Copy Status 0-11 into X field of C
ST=C		Copy X field of C into Status 0-11
CLRST		Clear Status 0-11

4.3.6.2 Hardware Status

SB=0	Clear Sticky Bit
SR=0	Clear Service Request bit (see SREQ?)
MP=0	Clear Module-Pulled bit
XM=0	Clear External Module Missing bit
CLRHST	Clear all 4 Hardware Status bits

4.3.7 System Control

SETHEX	Set arithmetic mode to hexadecimal
SETDEC	Set arithmetic mode to decimal
SREQ?	Sets Service Request bit if service has has been requested. C(0) shows what bit(s) are pulled high (if any)
C=RSTK	Pop return stack into A-field of C
RSTK=C	Push A-field of C onto return stack
CONFIG	Configure
UNCNFG	Unconfigure
RESET	Send Reset command to system bus
BUSCC	Send Bus command C onto system bus
SHUTDN	Stop CPU here (sleeps until wake-up)
C=ID	Request chip ID into A-field of C
INTOFF	Disable interrupts (doesn't affect ON-key or module-pulled interrupts)
INTON	Enable interrupts

4.3.8 Keyscan Instructions

OUT=C	Copy X field of C to OUTput register
OUT=CS	Copy nibble 0 of C to OUTput register
A=IN	Copy INput register to lower 4 nibbles of A
C=IN	Copy INput register to lower 4 nibbles of C

4.3.9 Register Swaps

```
----
 s = R0,R1,R2,R3,R4
----
```

AsEX	Exchange register A with s
CsEX	Exchange register C with s
A=s	Copy s to register A
C=s	Copy s to register C
s=A	Copy register A to s
s=C	Copy register C to s

4.3.10 Data Pointer Manipulation

```
----
 d = D0,D1
```

```
 1 <= n <= 16
 expr <= 5 nibbles
----
```

AdEX		Exchange Data ptr d with A-field of A
CdEX		Exchange Data ptr d with A-field of C
AdXS		Exchange lower 4 nibs of Data ptr d with lower 4 nibs of A
CdXS		Exchange lower 4 nibs of Data ptr d with lower 4 nibs of C
d=A		Copy A-field of A to Data pointer d
d=C		Copy A-field of C to Data pointer d
d=AS		Copy lower 4 nibs of A to lower 4 nibs of Data pointer d
d=CS		Copy lower 4 nibs of C to lower 4 nibs of Data pointer d
d=d+	n	Increment Data pointer d by n
d=d-	n	Decrement Data pointer d by n
d=HEX	hh	Load hh into lower 2 nibs of Data ptr d
d=HEX	hhhh	Load hhhh into lower 4 nibs of Data ptr d
d=HEX	hhhhh	Load hhhhh into Data ptr d
d=(2)	nn	Load nn into lower 2 nibs of Data ptr d (any overflow is ignored)
d=(4)	nnnn	Load nnnn into lower 4 nibs of Data ptr d (any overflow is ignored)
d=(5)	nnnnn	Load nnnnn into Data ptr d (any overflow is ignored)

4.3.11 Data Transfer

```
----
 fsd = Field select fs, or d (# of digits). If d, then
       the copy starts at nibble 0 of the working register.
 1 <= d <= 16
----
```

A=DAT0 fsd	Copy data from memory addressed by D0 into A, field selected
C=DAT0 fsd	Copy data from memory addressed by D0 into C, field selected
A=DAT1 fsd	Copy data from memory addressed by D1 into A, field selected
C=DAT1 fsd	Copy data from memory addressed by D1 into C, field selected
DAT0=A fsd	Copy data from A into memory addressed by D0, field selected
DAT0=C fsd	Copy data from C into memory addressed by D0, field selected
DAT1=A fsd	Copy data from A into memory addressed by

```
                                 D1, field selected
      DAT1=C fsd                 Copy data from C into memory addressed by
                                   D1, field selected
```

4.3.12 Load Constants

```
      LCHEX  hhhhhhhh  Load hex constant into C (1 to 16 digits)
      LC(m)  expr      Load the m-nibble constant into C
      LCASC  'ascii'   Load up to 8 ASCII characters into C
      LCASC  \ascii\   Load up to 8 ASCII characters into C
```

4.3.13 Shift Instructions

Note that right shifts (circular or non-circular) will set the Sticky Bit (SB=1) if a nonzero bit is shifted off to the right. Otherwise, SB is unchanged.

```
      ----
       r = A,B,C,D
       fs = Field Select
      ----

      rSL    fs        Shift register r fs field Left  1 nibble
      rSR    fs        Shift register r fs field Right 1 nibble
      rSLC             Shift register r Left  Circular 1 nibble
      rSRC             Shift register r Right Circular 1 nibble
      rSRB             Shift register r Right          1 bit
```

4.3.14 Logical Operations

Logical operations are bit-wise.

```
      ----
       r,s = A,B,C  or  (r,s) = (C,D),(D,C)
       fs = Field Select
      ----

      r=r&s  fs        r AND s into r, field selected
      r=r!s  fs        r OR  s into r, field selected
```

4.3.15 Arithmetics

The two groups of arithmetics differ in the range of registers available. In the first group (General usage) almost all combinations of the four working registers are possible; however,

in the second group (Restricted usage) only a few select combinations are possible.

4.3.15.1 General Usage

```
----
 r,s = A,B,C  or (r,s) = (C,D),(D,C)
 fs  = Field Select
----

r=0    fs      Set r to zero
r=r+r  fs      Double r, adjust Carry
r=r+1  fs      Increment r by 1, adjust Carry
r=r-1  fs      Decrement r by 1, adjust Carry
r=-r   fs      10'S complement or 2'S complement, Carry
                 set if r#0, else clear
r=-r-1 fs      9'S complement or 1'S complement
                 Carry always cleared
r=r+s  fs      Sum r and s into r, adjust Carry
s=r+s  fs      Sum r and s into s, adjust Carry
r=s    fs      Copy s into r
s=r    fs      Copy r into s
rsEX   fs      Exchange r and s
```

4.3.15.2 Restricted Usage

```
----
 (r,s) = (A,B),(B,C),(C,A),(D,C)
----

r=r-s  fs      Difference of r and s into r, adjust Carry
r=s-r  fs      Difference of s and r into r, adjust Carry
s=s-r  fs      Difference of s and r into s, adjust Carry
```

4.3.16 No-Op Instructions

Execution of a No-Op affects no CPU registers except for the PC.

```
NOP3           Three nibble No-Op
NOP4           Four nibble No-Op
NOP5           Five nibble No-Op
```

4.3.17 Pseudo-Ops

4.3.17.1 Data Storage Allocation

```
----
1 <= n <= 8
----
```

BSS	nnnnn	Allocate nnnnn number of zero nibs
CON(m)	expr	Generate m-nibble constant (digits are reversed in the opcode)
REL(m)	expr	Generate m-nibble relative constant (digits reversed in the opcode)
NIBASC	'ascii'	Generate ascii characters, byte reversed
NIBASC	\ascii\	Generate ascii characters, byte reversed
NIBHEX	hhhh	Generate hexadecimal digits hhhh (digits are not reversed in the opcode)

4.3.17.2 Conditional Assembly

name	IF	expr	Start conditional assembly until ELSE or ENDIF if flag expr was set on invocation of assembler (optional use of name allows nesting of IF's)
name	ELSE		Conditional assembly if IF test was false
name	ENDIF		Ends conditional assembly started by IF

4.3.17.3 Listing Formatting

EJECT		Force new page in the assembly listing
STITLE	text	Force new page, set subtitle value to text
TITLE	text	Set title value to text

4.3.17.4 Symbol Definition

label	EQU	nnnnn	Defines label to have the value expr

4.3.17.5 Assembly Mode

ABS	nnnnn	Specify absolute assembly at adress given
END		Marks end of the assembly source

4.4 Mnemonic Dictionary

This section contains a description of each HP-71 assembler instruction or pseudo-op. The description shows the binary opcode generated by the mnemonic, if any, as well as the execution cycle time required if the mnemonic is an executable instruction.

The symbols used in these descriptions are explained in the "Explanation of Symbols" section earlier in this chapter.

```
**********************************************************
*                       MNEMONICS                        *
**********************************************************
```

?A#0 fs - Test for A not equal to 0

```
fs = A                          opcode:  8ACyy
                                cycles:   13 + d (GO/RTNYES)
                                           6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W)        opcode:  9aCyy
                                cycles:   13 + d (GO/RTNYES)
                                           6 + d (NO)
```

Test whether the fs field of A is not equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A#B fs - Test for A not equal to B

```
fs = A                          opcode:  8A4yy
                                cycles:   13 + d (GO/RTNYES)
                                           6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W)        opcode:  9a4yy
                                cycles:   13 + d (GO/RTNYES)
```

6 + d (NO)

Test whether the fs field of A is not equal to the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A#C fs - Test for A not equal to C

---

fs = A

opcode: 8A6yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W)

opcode: 9a6yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of A is not equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A<=B fs - Test for A less than or equal to B

---

fs = A

opcode: 8BCyy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W)

opcode: 9bCyy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of A is less than or equal to the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A<B fs - Test for A less than B

fs = A — opcode: 8B4yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9b4yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of A is less than the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A=0 fs - Test for A equal to 0

fs = A — opcode: 8A8yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a8yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of A is equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A=B fs - Test for A equal to B

fs = A — opcode: 8A0yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a0yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of A is equal to the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A=C fs - Test for A equal to C

fs = A — opcode: 8A2yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a2yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of A is equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A>=B fs - Test for A greater than or equal to B

fs = A — opcode: 8B8yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9b8yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of A is greater than or equal to the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?A>B fs - Test for A greater than B

fs = A — opcode: 8B0yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9b0yy
cycles: 13 + d (GO/RTNYES)

6 + d (NO)

Test whether the fs field of A is greater than the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B#0 fs - Test for B not equal to 0

---

fs = A — opcode: 8ADyy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9?Dyy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of B is not equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B#A fs - Test for B not equal to A

---

fs = A — opcode: 3A4yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a4yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of B is not equal to the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B#C fs - Test for B not equal to C

---

fs = A	opcode:	8A5yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a5yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is not equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B<=C fs - Test for B less than or equal to C

---

fs = A	opcode:	8BDyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9bDyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is less than or equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B<C fs - Test for B less than C

---

fs = A	opcode:	8B5yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9b5yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is less than the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B=0 fs - Test for B equal to 0

fs = A	opcode:	8A9yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a9yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B=A fs - Test for B equal to A

fs = A	opcode:	8A0yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a0yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is equal to the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B=C fs - Test for B equal to C

fs = A	opcode:	8A1yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a1yy
	cycles:	13 + d (GO/RTNYES)

6 + d (NO)

Test whether the fs field of B is equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B>=C fs - Test for B greater than or equal to C

---

fs = A	opcode:	8B9yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9b9yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is greater than or equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?B>C fs - Test for B greater than C

---

fs = A	opcode:	8B1yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9b1yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of B is greater than the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C#0 fs - Test for C not equal to 0

fs = A — opcode: 8AEyy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9aEyy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of C is not equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C#A fs - Test for C not equal to A

fs = A — opcode: 8A6yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a6yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of C is nct equal to the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C#B fs - Test for C not equal to B

fs = A — opcode: 8A5yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a5yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of C is not equal to the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C#D fs - Test for C not equal to D

fs = A	opcode:	8A7yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a7yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of C is not equal to the fs field of D. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C<=A fs - Test for C less than or equal to A

fs = A	opcode:	8BEyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9bEyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of C is less than or equal to the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C<A fs - Test for C less than A

fs = A	opcode:	8B6yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9b6yy
	cycles:	13 + d (GO/RTNYES)

6 + d (NO)

Test whether the fs field of C is less than the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C=0 fs - Test for C equal to 0

---

fs = A	opcode:	8AAyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9aAyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of C is equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C=A fs - Test for C equal to A

---

fs = A	opcode:	8A2yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a2yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of C is equal to the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C=B fs - Test for C equal to B

fs = A — opcode: 8A1yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a1yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of C is equal to the fs field of B. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C=D fs - Test for C equal to D

fs = A — opcode: 8A3yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a3yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of C is equal to the fs field of D. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C>=A fs - Test for C greater than or equal to A

fs = A — opcode: 8BAyy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9bAyy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of C is greater than or equal to the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?C>A fs - Test for C greater than A

fs = A	opcode:	8B2yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9b2yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of C is greater than the fs field of A. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D#0 fs - Test for D not equal to 0

fs = A	opcode:	8AFyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9aFyy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)

Test whether the fs field of D is not equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D#C fs - Test for D not equal to C

fs = A	opcode:	8A7yy
	cycles:	13 + d (GO/RTNYES)
		6 + d (NO)
fs = (P,WP,XS,X,S,M,B,W)	opcode:	9a7yy
	cycles:	13 + d (GO/RTNYES)

6 + d (NO)

Test whether the fs field of D is not equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D<=C fs - Test for D less than or equal to C

---

fs = A

opcode: 8BFyy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W)

opcode: 9bFyy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of D is less than or equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D<C fs - Test for D less than to C

---

fs = A

opcode: 8B7yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

= (P,WP,XS,X,S,M,B,W)

opcode: 9b7yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of D is less than the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D=0 fs - Test for D equal to 0

fs = A — opcode: 8AByy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9aByy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of D is equal to 0. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D=C fs - Test for D equal to C

fs = A — opcode: 8A3yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9a3yy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of D is equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D>=C fs - Test for D greater than or equal to C

fs = A — opcode: 8BByy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W) — opcode: 9bByy; cycles: 13 + d (GO/RTNYES), 6 + d (NO)

Test whether the fs field of D is greater than or equal to the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?D>C fs - Test for D greater than C

fs = A

opcode: 8B3yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

fs = (P,WP,XS,X,S,M,B,W)

opcode: 9b3yy
cycles: 13 + d (GO/RTNYES)
6 + d (NO)

Test whether the fs field of D is greater than the fs field of C. Must be followed by a GOYES or RTNYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?MP=0 - Test Module Pulled bit (MP)

opcode: 838yy
cycles: 13 (GO/RTNYES)
6 (NO)

Test whether the Module Pulled bit (MP) is zero. This hardware status bit is set whenever a module-pulled interrupt occurs (the *INT line of the CPU is pulled high), and must be explictly cleared by the MP=0 mnemonic. See the "HP-71 Hardware Specification" for more information. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?P# n - Test if P pointer not equal to n

opcode: 88nyy
cycles: 13 (GO/RTNYES)
6 (NO)

Test whether the P pointer is not equal to n. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?P= n - Test if P pointer is equal to n

```
                          opcode:  89nyy
                          cycles:   13 (GO/RTNYES)
                                     6 (NO)
```

Test whether the P pointer is equal to n. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?SB=0 - Test Sticky Bit (SB)

```
                          opcode:  832yy
                          cycles:   13 (GO/RTNYES)
                                     6 (NO)
```

Test whether the Sticky Bit (SB) is zero. This hardware status bit is set on right shifts (circular or non-circular) when a non-zero nibble or bit is shifted off the end of the field. The Sticky Bit must be cleared explicitly by the SB=0 mnemonic. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?SR=0 - Test Service Request bit (SR) for zero

```
                          opcode:  834yy
                          cycles:   13 (GO/RTNYES)
                                     6 (NO)
```

Test whether the Service Request bit (SR) is zero. This hardware status bit is set by the SREQ? mnemonic, and must be cleared explicitly by the SR=0 mnemonic. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?ST#0 n - Test status bit n not equal to 0

opcode: 87nyy
cycles: 14 (GO/RTNYES)
7 (NO)

Test whether Program Status bit n is set. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?ST#1 n - Test status bit n not equal to 1

opcode: 86nyy
cycles: 14 (GO/RTNYES)
7 (NO)

Test whether Program Status bit n is clear. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?ST=0 n - Test status bit n equal to 0

opcode: 86nyy
cycles: 14 (GO/RTNYES)
7 (NO)

Test whether Program Status bit n is clear. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?ST=1 n - Test status bit n equal to 1

opcode: 87nyy
cycles: 14 (GO/RTNYES)
7 (NO)

Test whether Program Status bit n is set. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

?XM=0 - Test External Module Missing bit (XM)

opcode: 831yy
cycles: 13 (GO/RTNYES)
6 (NO)

Test the whether the External Module Missing bit (XM) is zero. This hardware status bit is set by the RTNSXM mnemonic, and must be explicitly cleared by the XM=0 mnemonic. Must be followed by a RTNYES or GOYES mnemonic. yy is determined by the following RTNYES or GOYES. Adjusts Carry.

A=-A fs - Two's complement of A into A

fs = A
opcode: F8
cycles: 7

fs = (P,WP,XS,X,S,M,B,W)
opcode: Bb8
cycles: 3 + d

Complement the specified fs field of A. Complement is two's complement if in HEX mode, ten's complement if in DEC mode. Carry is set if the field is not zero, else Carry is cleared.

A=-A-1 fs - One's complement of A into A

---

fs = A	opcode: FC cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BbC cycles: 3 + d

Perform a one's complement on the specified fs field of A. Carry is always cleared.

A=0 fs - Set A equal to 0

---

fs = A	opcode: D0 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ab0 cycles: 3 + d

Set the specified fs field of A to zero. Carry is not affected.

A=A!B fs - A OR B into A

---

fs = A	opcode: 0FF8 cycles: 4 + d
fs = (P,WP,XS,X,S,M,B,W)	opcode: 0Ea8 cycles: 4 + d

Set the fs field of register A to its logical OR with the corresponding field of register B. Carry is not affected.

A=A!C fs -- A OR C into A

fs = A	opcode:	0EFE
	cycles:	4 + d
fs = (P,WP,XS,X,S,M,B,W)	opcode:	0EaE
	cycles:	4 + d

Set the fs field of register A to its logical OR with the corresponding field of register C. Carry is not affected.

A=A&B fs - A AND B into A

fs = A	opcode:	0EF0
	cycles:	4 + d
fs = (P,WP,XS,X,S,M,B,W)	opcode:	0Ea0
	cycles:	4 + d

Set the fs field of register A to its logical AND with the corresponding field of register B. Carry is not affected.

A=A&C fs - A AND C into A

fs = A	opcode:	0EF6
	cycles:	4 + d
fs = (P,WP,XS,X,S,M,B,W)	opcode:	0Ea6
	cycles:	4 + d

Set the fs field of register A to its logical AND with the corresponding field of register C. Carry is not affected.

A=A+1 fs - Increment A

fs = A	opcode: E4 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba4 cycles: 3 + d

Increment the specified fs field of register A by one. Adjusts Carry.

A=A+A fs - Sum of A and A into A

fs = A	opcode: C4 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Aa4 cycles: 3 + d

Double the specified fs field of register A. Adjusts Carry.

A=A+B fs - Sum of A and B into A

fs = A	opcode: C0 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Aa0 cycles: 3 + d

Set the specified fs field of register A to the sum of itself and the corresponding field of register B. Adjusts Carry.

A=A+C fs - Sum of A and C into A

fs = A	opcode: CA cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AaA cycles: 3 + d

Set the specified fs field of register A to the sum of itself and the corresponding field of register C. Adjusts Carry.

A=A-1 fs - Decrement A

fs = A	opcode: CC cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AaC cycles: 3 + d

Decrement the specified fs field of register A by one. Adjusts Carry.

A=A-B fs - A minus B into A

fs = A	opcode: E0 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba0 cycles: 3 + d

Set the specified fs field of register A to the difference between itself and the corresponding field of register B. Adjusts Carry.

A=A-C fs - A minus C into A

fs = A	opcode: EA cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BaA cycles: 3 + d

Set the specified fs field of register A to the difference between itself and the corresponding field of register C. Adjusts Carry.

A=B fs - Copy B to A

fs = A	opcode: D4 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ab4 cycles: 3 + d

Copy the fs field of register B into the corresponding field of register A. Carry is not affected.

A=B-A fs - B minus A into A

fs = A	opcode: EC cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BaC cycles: 3 + d

Set the specified fs field of register A to the inverse difference between itself and the corresponding field of register B. Adjusts Carry.

A=C fs - Copy C to A

fs = A	opcode: DA cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AbA cycles: 3 + d

Copy the fs field of register C into the corresponding field of register A. Carry is not affected.

A=DAT0 fsd - Load A from memory

fs = A	opcode: 142 cycles: 18
fs = B	opcode: 14A cycles: 15
fs = (P,WP,XS,X,S,M,W)	opcode: 152a cycles: 17 + d
fs = d	opcode: 15Ax (x=d-1) cycles: 16 + d

The amount of data (d nibbles) specified by fsd will be transferred from the memory address pointed to by D0 into the specified field of register A. The lowest-addressed nibble will be transferred into the lowest-order nibble of the register field, proceeding toward the higher-order nibbles. If fs = d, d nibbles are transferred into the register starting at nibble 0. See the section on "Loading Data From Memory" earlier in this chapter.

A=DAT1 fsd - Load A from memory

fs = A	opcode: 143 cycles: 18
fs = B	opcode: 14B cycles: 15

fs = (P,WP,XS,X,S,M,W) opcode: 153a
                        cycles: 17 + d

fs = d   opcode: 15Bx (x=d-1)
                        cycles: 16 + d

The amount of data (d nibbles) specified by fsd will be transferred from the memory address pointed to by D1 into the specified field of register A. The lowest-addressed nibble will be transferred into the lowest-order nibble of the register field, proceeding toward the higher-order nibbles. If fs = d, d nibbles are transferred into the register starting at nibble 0. See the section on "Loading Data From Memory" earlier in this chapter.

A=IN - Load A with IN
---

opcode: 802
cycles: 7

Load the low-order 4 nibbles of the A register with the contents of the Input register.

A=R0 - Copy R0 to A
---

opcode: 110
cycles: 19

The contents of the scratch register R0 is copied to the working register A.

A=R1 - Copy R1 to A

opcode: 111
cycles: 19

The contents of the scratch register R1 is copied to the working register A.

A=R2 - Copy R2 to A

opcode: 112
cycles: 19

The contents of the scratch register R2 is copied to the working register A.

A=R3 - Copy R3 to A

opcode: 113
cycles: 19

The contents of the scratch register R3 is copied to the working register A.

A=R4 - Copy R4 to A

opcode: 114
cycles: 19

The contents of the scratch register R4 is copied to the working register A.

ABEX fs - Exchange Registers A and B

fs = A — opcode: DC, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: AbC, cycles: 3 + d

Exchange the fs fields of registers of A and B. Carry is not affected.

ACEX fs - Exchange Registers A and C

fs = A — opcode: DE, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: AbE, cycles: 3 + d

Exchange the fs fields of registers of A and C. Carry is not affected.

ADOEX - Exchange A and D0 (nibs 0-4)

opcode: 132
cycles: 8

Exchange the A field of register A with Data pointer D0. Carry is not affected.

AD0XS - Exchange A and D0 short (nibs 0-3)

opcode: 13A
cycles: 7

Exchange the lower 4 nibbles of A with the lower 4 nibbles of Data pointer D0. Carry is not affected.

AD1EX - Exchange A and D1 (nibs 0-4)

opcode: 133
cycles: 8

Exchange the A field of register A with Data pointer D1. Carry is not affected.

AD1XS - Exchange A and D1 short (nibs 0-3)

opcode: 13B
cycles: 7

Exchange the lower 4 nibbles of A with the lower 4 nibbles of Data pointer D1. Carry is not affected.

AR0EX - Exchange A and R0

opcode: 120
cycles: 19

Exchange the contents of the working register A and the scratch register R0.

AR1EX -- Exchange A and R1

opcode: 121
cycles: 19

Exchange the contents of the working register A and the scratch register R1.

AR2EX - Exchange A and R2

opcode: 122
cycles: 19

Exchange the contents of the working register A and the scratch register R2.

AR3EX - Exchange A and R3

opcode: 123
cycles: 19

Exchange the contents of the working register A and the scratch register R3.

AR4EX - Exchange A and R4

opcode: 124
cycles: 19

Exchange the contents of the working register A and the scratch register R4.

ASL fs - A Shift Left

fs = A — opcode: F0, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: BbO, cycles: 3 + d

Shift the contents of the specified fs field of register A left one nibble, without affecting the rest of the register. The nibble shifted off the left end of the field is lost. The new low-order nibble of the field is zero. The Sticky Bit (SB) is not affected.

ASLC - A Shift Left Circular

opcode: 810
cycles: 21

Circular shift register A left one nibble. Operates on all 16 digits. The Sticky Bit (SB) is not affected.

ASR fs - A Shift Right

fs = A — opcode: F4, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Bb4, cycles: 3 + d

Shift the contents of the specified fs field of register A right one nibble, without affecting the rest of the register. The nibble shifted off the right end of the field is lost, but the Sticky Bit (SB) is set if the nibble was non-zero. The new high-order nibble of the field is zero.

ASRB - A Shift Right Bit

opcode: 81C
cycles: 20

Shift register A right one bit. Cperates on all 16 digits. The bit shifted off the end is lost, but the Sticky Bit (SB) is set if it was non-zero. The new high-order bit of the register is zero.

ASRC - A Shift Right Circular

opcode: 814
cycles: 21

Circular shift register A right one nibble. Operates on all 16 digits. The Sticky Bit (SB) is set if the nibble shifted from low-order around to high-order position was non-zero.

B=-B fs - Two's complement of B into B

fs = A opcode: F9
cycles: 7

fs = (P,WP,XS,X,S,M,B,W) opcode: Bb9
cycles: 3 + d

Complement the specified fs field of B. Complement is two's complement if in HEX mode, ten's complement if in DEC mode. Carry is set if the field is not zero, else Carry is cleared.

B=-B-1 fs - One's complement of B into B

fs = A	opcode: FD cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BbD cycles: 3 + d

Perform a one's complement on the specified fs field of B. Carry is always cleared.

B=0 fs - Set B equal to 0

fs = A	opcode: D1 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ab1 cycles: 3 + d

Set the specified fs field of B to zero. Carry is not affected.

B=A fs - Copy A to B

fs = A	opcode: D8 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ab8 cycles: 3 + d

Copy the fs field of register A into the corresponding field of register B. Carry is not affected.

B=B!A fs - B OR A into B

fs = A — opcode: 0EFC; cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0EaC; cycles: 4 + d

Set the fs field of register B to its logical OR with the corresponding field of register A. Carry is not affected.

B=B!C fs - B OR C into B

fs = A — opcode: 0EF9; cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0Ea9; cycles: 4 + d

Set the fs field of register B to its logical OR with the corresponding field of register C. Carry is not affected.

B=B&A fs - B AND A into B

fs = A — opcode: 0EF4; cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0Ea4; cycles: 4 + d

Set the fs field of register B to its logical AND with the corresponding field of register A. Carry is not affected.

B=B&C fs - B AND C into B

fs = A — opcode: 0EF1, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0Ea1, cycles: 4 + d

Set the fs field of register B to its logical AND with the corresponding field of register C. Carry is not affected.

B=B+1 fs - Increment B

fs = A — opcode: E5, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Ba5, cycles: 3 + d

Increment the specified fs field of register B by one. Adjusts Carry.

B=B+A fs - Sum of B and A into B

fs = A — opcode: C8, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Aa8, cycles: 3 + d

Set the specified fs field of register B to the sum of itself and the corresponding field of register A. Adjusts Carry.

B=B+B fs - Sum of B and B into B

fs = A	opcode:	C5
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Aa5
	cycles:	3 + d

Double the specified fs field of register B. Adjusts Carry.

B=B+C fs - Sum of B and C into B

fs = A	opcode:	C1
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Aa1
	cycles:	3 + d

Set the specified fs field of register B to the sum of itself and the corresponding field of register C. Adjusts Carry.

B=B-1 fs - Decrement B

fs = A	opcode:	CD
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	AaD
	cycles:	3 + d

Decrement the specified fs field of register B by one. Adjusts Carry.

B=B-A fs - B minus A into B

fs = A	opcode:	E8
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Ba8
	cycles:	3 + d

Set the specified fs field of register B to the difference between itself and the corresponding field of register A. Adjusts Carry.

B=B-C fs - B minus C into B

fs = A	opcode:	E1
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Ba1
	cycles:	3 + d

Set the specified fs field of register B to the difference between itself and the corresponding field of register C. Adjusts Carry.

B=C fs - Copy C to B

fs = A	opcode:	D5
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Ab5
	cycles:	3 + d

Copy the fs field of register C into the corresponding field of register B. Carry is not affected.

B=C-B fs - C minus B into B

fs = A	opcode: ED cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BaD cycles: 3 + d

Set the specified fs field of register B to the inverse difference between itself and the corresponding field of register C. Adjusts Carry.

BAEX fs - Exchange Registers B and A

fs = A	opcode: DC cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AbC cycles: 3 + d

Exchange the fs fields of registers of B and A. Carry is not affected.

BCEX fs - Exchange Registers B and C

fs = A	opcode: DD cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AbD cycles: 3 + d

Exchange the fs fields of registers of B and C. Carry is not affected.

BSL fs - B Shift Left

fs = A — opcode: F1, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Bb1, cycles: 3 + d

Shift the contents of the specified fs field of register B left one nibble, without affecting the rest of the register. The nibble shifted off the left end of the field is lost. The new low-order nibble of the field is zero. The Sticky Bit (SB) is not affected.

BSLC - B Shift Left Circular

opcode: 811
cycles: 21

Circular shift register B left one nibble. Operates on all 16 digits. The Sticky Bit (SB) is not affected.

BSR fs - B Shift Right

fs = A — opcode: F5, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Bb5, cycles: 3 + d

Shift the contents of the specified fs field of register B right one nibble, without affecting the rest of the register. The nibble shifted off the right end of the field is lost, but the Sticky Bit (SB) is set if the nibble was non-zero. The new high-order nibble of the field is zero.

BSRB - B Shift Right Bit

opcode: 81D
cycles: 20

Shift register B right one bit. Operates on all 16 digits. The bit shifted off the end is lost, but the Sticky Bit (SB) is set if it was non-zero. The new high-order bit of the register is zero.

BSRC - B Shift Right Circular

opcode: 815
cycles: 21

Circular shift register B right one nibble. Operates on all 16 digits. The Sticky Bit (SB) is set if the nibble shifted from low-order around to high-order position was non-zero.

BUSCC - Bus Command "C"

opcode: 80B
cycles: 6

Enters the HP-71 bus command "C" onto the system bus (this command is reserved for later use). No other operation is performed. See the "HP-71 Hardware Specification" for more information.

C+P+1 - Increment C by One Plus P Pointer

opcode: 809
cycles: 8

The A field of the C register is incremented by one plus the value of the P pointer. This instruction is always executed in HEX mode. Adjusts Carry.

C=-C fs - Two's complement of C into C

fs = A	opcode:	FA
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	BbA
	cycles:	3 + d

Complement the specified fs field of C. Complement is two's complement if in HEX mode, ten's complement if in DEC mode. Carry is set if the field is not zero, else Carry is cleared.

C=-C-1 fs - One's complement of C into C

fs = A	opcode:	FE
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	BbE
	cycles:	3 + d

Perform a one's complement on the specified fs field of C. Carry is always cleared.

C=0 fs - Set C equal to 0

fs = A	opcode:	D2
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Ab2
	cycles:	3 + d

Set the specified fs field of C to zero. Carry is not affected.

C=A fs - Copy A to C

fs = A — opcode: D6, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Ab6, cycles: 3 + d

Copy the fs field of register A into the corresponding field of register C. Carry is not affected.

C=A-C fs - A minus C into C

fs = A — opcode: EE, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: BaE, cycles: 3 + d

Set the specified fs field of register C to the inverse difference between itself and the corresponding field of register A. Adjusts Carry.

C=B fs - Copy B to C

fs = A — opcode: D9, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Ab9, cycles: 3 + d

Copy the fs field of register B into the corresponding field of register C. Carry is not affected.

C=C!A fs - C OR A into C

fs = A — opcode: 0EFA, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0EaA, cycles: 4 + d

Set the fs field of register C to its logical OR with the corresponding field of register A. Carry is not affected.

C=C!B fs - C OR B into C

fs = A — opcode: 0EFD, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0EaD, cycles: 4 + d

Set the fs field of register C to its logical OR with the corresponding field of register B. Carry is not affected.

C=C!D fs - C OR D into C

fs = A — opcode: 0EFF, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0EaF, cycles: 4 + d

Set the fs field of register C to its logical OR with the corresponding field of register D. Carry is not affected.

C=C&A fs - C AND A into C

fs = A — opcode: 0EF2, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0Ea2, cycles: 4 + d

Set the fs field of register C to its logical AND with the corresponding field of register A. Carry is not affected.

C=C&B fs - C AND B into C

fs = A — opcode: 0EF5, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0Ea5, cycles: 4 + d

Set the fs field of register C to its logical AND with the corresponding field of register B. Carry is not affected.

C=C&D fs - C AND D into C

fs = A — opcode: 0EF7, cycles: 4 + d

fs = (P,WP,XS,X,S,M,B,W) — opcode: 0Ea7, cycles: 4 + d

Set the fs field of register C to its logical AND with the corresponding field of register D. Carry is not affected.

C=C+1 fs - Increment C

fs = A	opcode: E6 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba6 cycles: 3 + d

Increment the specified fs field of register C by one. Adjusts Carry.

C=C+A fs - Sum of C and A into C

fs = A	opcode: C2 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Aa2 cycles: 3 + d

Set the specified fs field of register C to the sum of itself and the corresponding field of register A. Adjusts Carry.

C=C+B fs - Sum of C and B into C

fs = A	opcode: C9 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Aa9 cycles: 3 + d

Set the specified fs field of register C to the sum of itself and the corresponding field of register B. Adjusts Carry.

C=C+C fs - Sum of C and C into C

fs = A	opcode:	C6
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	Aa6
	cycles:	3 + d

Double the specified fs field of register C. Adjusts Carry.

C=C+D fs - Sum of C and D into C

fs = A	opcode:	CB
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	AaB
	cycles:	3 + d

Set the specified fs field of register C to the sum of itself and the corresponding field of register D. Adjusts Carry.

C=C-1 fs - Decrement C

fs = A	opcode:	CE
	cycles:	7
fs = (P,WP,XS,X,S,M,B,W)	opcode:	AaE
	cycles:	3 + d

Decrement the specified fs field of register C by one. Adjusts Carry.

C=C-A fs - C minus A into C

fs = A	opcode: E2 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba2 cycles: 3 + d

Set the specified fs field of register C to the difference between itself and the corresponding field of register A. Adjusts Carry.

C=C-B fs - C minus B into C

fs = A	opcode: E9 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba9 cycles: 3 + d

Set the specified fs field of register C to the difference between itself and the corresponding field of register B. Adjusts Carry.

C=C-D fs - C minus D into C

fs = A	opcode: EB cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BaB cycles: 3 + d

Set the specified fs field of register C to the difference between itself and the corresponding field of register D. Adjusts Carry.

C=D fs - Copy D to C

---

fs = A	**opcode: DB**
	cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	**opcode: AbB**
	cycles: 3 + d

Copy the fs field of register D into the corresponding field of register C. Carry is not affected.

C=DAT0 fsd - Load C from memory

---

fs = A	opcode: 146
	cycles: 18
fs = B	opcode: 14E
	cycles: 15
fs = (P,WP,XS,X,S,M,W)	opcode: 156a
	cycles: 17 + d
fs = d	opcode: 15Ex (x=d-1)
	cycles: 16 + d

The amount of data (d nibbles) specified by fsd will be transferred from the memory address pointed to by D0 into the specified field of register C. The lowest-addressed nibble will be transferred into the lowest-order nibble of the register field, proceeding toward the higher-order nibbles. If fs = d, d nibbles are transferred into the register starting at nibble 0. See the section on "Loading Data From Memory" earlier in this chapter.

C=DAT1 fsd - Load C from memory

fs = A	**opcode: 147** **cycles: 18**
fs = B	**opcode: 14F** **cycles: 15**
fs = (P,WP,XS,X,S,M,W)	**opcode: 157a** **cycles: 17 + d**
fs = d	**opcode: 15Fx (x=d-1)** **cycles: 16 + d**

The amount of data (d nibbles) specified by fsd will be transferred from the memory address pointed to by D1 into the specified field of register C. The lowest-addressed nibble will be transferred into the lowest-order nibble of the register field, proceeding toward the higher-order nibbles. If fs = d, d nibbles are transferred into the register starting at nibble 0. See the section on "Loading Data From Memory" earlier in this chapter.

C=ID - Request chip ID

opcode: 806
cycles: 11

The chip which has its DAISY-IN line high and its configuration flag low will send its 5 nibble ID register to the system bus which will be loaded into the low-order 5 nibbles (A field) of the C register. See the "HP-71 Hardware Specification" for more information.

C=IN - Load C with IN

opcode: 803
cycles: 7

Load the low-order 4 nibbles of the C register with the contents of the Input register. See the "HP-71 Hardware Specification" for more information.

C=P n - Copy P Pointer into Nibble n of C

opcode: 80Cn
cycles: 6

Copy P pointer into C register at digit position specified by n.

C=R0 - Copy R0 to C

opcode: 118
cycles: 19

The contents of the scratch register R0 is copied to the working register C.

C=R1 - Copy R1 to C

opcode: 119
cycles: 19

The contents of the scratch register R1 is copied to the working register C.

C=R2 - Copy R2 to C

opcode: 11A
cycles: 19

The contents of the scratch register R2 is copied to the working register C.

C=R3 - Copy R3 to C

opcode: 11B
cycles: 19

The contents of the scratch register R3 is copied to the working register C.

C=R4 - Copy R4 to C

opcode: 11C
cycles: 19

The contents of the scratch register R4 is copied to the working register C.

C=RSTK - Pop stack to C

opcode: 07
cycles: 8

Pop the top-most address off of the hardware return stack, placing the address in the lower 5 nibbles (A field) of register C. The high-order nibbles of C are unchanged. As the address is popped from the return stack, a zero address is inserted at the bottom of the stack. Compare with the RTN mnemonic.

C=ST - Status to C

opcode: 09
cycles: 6

Copy the low-order 12 bits of the status register into the low-order 12 bits (X field) of the C register.

CAEX fs - Exchange Registers C and A

fs = A — opcode: DE, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: AbE, cycles: 3 + d

Exchange the fs fields of registers of C and A. Carry is not affected.

CBEX fs - Exchange Registers C and B

fs = A — opcode: DD, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: AbD, cycles: 3 + d

Exchange the fs fields of registers of C and B. Carry is not affected.

CD0EX - Exchange C and D0 (nibs 0-4)

opcode: 136
cycles: 8

Exchange the A field of register C with Data pointer D0. Carry is not affected.

CD0XS - Exchange C and D0 short (nibs 0-3)

opcode: 13E
cycles: 7

Exchange the lower 4 nibbles of C with the lower 4 nibbles of Data pointer D0. Carry is not affected.

CD1EX - Exchange C and D1 (nibs 0-4)

opcode: 137
cycles: 8

Exchange the A field of register C with Data pointer D1. Carry is not affected.

CD1XS - Exchange C and D1 short (nibs 0-3)

opcode: 13F
cycles: 7

Exchange the lower 4 nibbles of C with the lower 4 nibbles of Data pointer D1. Carry is not affected.

CDEX fs - Exchange Registers C and D

---

fs = A opcode: DF
cycles: 7

fs = (P,WP,XS,X,S,M,B,W) opcode: AbF
cycles: 3 + d

Exchange the fs fields of registers of C and D. Carry is not affected.

CLRHST - Clear Hardware Status bits

---

opcode: 82F
cycles: 3

Clears the 4 Hardware Status bits XM, SB, SR and MP. Note that the opcode is actually 82x, where x is merely a mask for which Hardware Status bits to clear, as follows:

bit 0 - External Module Missing bit (see XM=0 mnemonic)
bit 1 - Sticky Bit (see SB=0 mnemonic)
bit 2 - Service Request bit (see SR=0 mnemonic)
bit 3 - Module Pulled bit (see MP=0 mnemonic)

For example opcode 829 clears XM and MP. Although there is no mnemonic for this, the opcode can be inserted into the code by using, for example, NIBHEX 829.

CLRST - Clear Program Status

---

opcode: 08
cycles: 6

Clear the low-order 12 bits (S0 through S11) of the Program Status register ST.

CONFIG - Configure

opcode: 805
cycles: 11

Copy the low-order 5 nibbles (A field) of the C register into the Configuration register of the chip which has its DAISY-IN line high and its configuration flag low. See the "HP-71 Hardware Specification" for information.

CPEX n - Exchange Nibble n of C With P Pointer

opcode: 80Fn
cycles: 6

Exchange the P pointer with digit n of the C register.

CR0EX - Exchange C and R0

opcode: 128
cycles: 19

Exchange the contents of the working register C and the scratch register R0.

CR1EX - Exchange C and R1

opcode: 129
cycles: 19

Exchange the contents of the working register C and the scratch register R1.

CR2EX - Exchange C and R2

opcode: 12A
cycles: 19

Exchange the contents of the working register C and the scratch register R2.

CR3EX - Exchange C and R3

opcode: 12B
cycles: 19

Exchange the contents of the working register C and the scratch register R3.

CR4EX - Exchange C and R4

opcode: 12C
cycles: 19

Exchange the contents of the working register C and the scratch register R4.

CSL fs - C Shift Left

fs = A — opcode: F2, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Bb2, cycles: 3 + d

Shift the contents of the specified fs field of register C left one nibble, without affecting the rest of the register. The nibble shifted off the left end of the field is lost. The new low-order nibble of the field is zero. The Sticky Bit (SB) is not affected.

CSLC - C Shift Left Circular

opcode: 812
cycles: 21

Circular shift register C left one nibble. Operates on all 16 digits. The Sticky Bit (SB) is not affected.

CSR fs - C Shift Right

fs = A opcode: F6
cycles: 7

fs = (P,WP,XS,X,S,M,B,W) opcode: Bb6
cycles: 3 + d

Shift the contents of the specified fs field of register C right one nibble, without affecting the rest of the register. The nibble shifted off the right end of the field is lost, but the Sticky Bit (SB) is set if the nibble was non-zero. The new high-order nibble of the field is zero.

CSRB - C Shift Right Bit

opcode: 81E
cycles: 20

Shift register C right one bit. Operates on all 16 digits. The bit shifted off the end is lost, but the Sticky Bit (SB) is set if it was non-zero. The new high-order bit of the register is zero.

CSRC - C Shift Right Circular

opcode: 816
cycles: 21

Circular shift register C right one nibble. Operates on all 16 digits. The Sticky Bit (SB) is set if the nibble shifted from low-order around to high-order position was non-zero.

CSTEX - Exchange Status with C register

opcode: 0B
cycles: 6

Exchange the low-order 12 bits (S0 through S11) of the Program Status register ST with the low-order 12 bits (X field) of the C register.

D0=(2) nn - Load 2 Nibbles Into D0

opcode: 19nn
cycles: 4

Load the low-order two nibbles of D0 with nn. The upper nibbles of D0 remain unchanged. Any overflow is ignored by the assembler. The assembled digits of nn are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D0=(4) nnnn - Load 4 Nibbles Into D0

opcode: 1Annnn
cycles: 6

Load the low-order four nibbles of D0 with nnnn. The upper nibble of D0 remains unchanged. Any overflow is ignored by the assembler. The assembled digits of nnnn are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D0=(5) nnnnn - Load 5 Nibbles Into D0

opcode: 1Bnnnnn
cycles: 7

Load all five nibbles of D0 with nnnnn. Any overflow is ignored by the assembler. The assembled digits of nnnnn are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D0=A - Copy A to D0 (nibs 0-4)

opcode: 130
cycles: 8

The A field of register A is copied into Data pointer register D0. Carry is not affected.

DO=AS - Copy A to DO short (nibs 0-3)

opcode: 138
cycles: 7

The lower 4 nibbles of A are copied into the lower 4 nibbles of Data pointer register DO. Carry is not affected.

DO=C - Copy C to DO (nibs 0-4)

opcode: 134
cycles: 8

The A field of register C is copied into Data pointer register DO. Carry is not affected.

DO=CS - Copy C to DO short (nibs 0-3)

opcode: 13C
cycles: 7

The lower 4 nibbles of C are copied into the lower 4 nibbles of Data pointer register DO. Carry is not affected.

DO=DO+ n - Add n to DO (1<=n<=16)

opcode: 16x (x=n-1)
cycles: 7

Increment DO by n. This instruction is always executed in HEX mode. Adjusts Carry.

D0=D0- n - Subtract n from D0 (1<=n<=16)

opcode: 18x (x=n-1)
cycles: 7

Decrement D0 by n. This instruction is always executed in HEX mode. Adjusts Carry.

D0=HEX hh - Load D0 with hex constant hh

opcode: 19hh
cycles: 4

Load the low-order two nibbles of D0 with the hex constant hh. The upper nibbles of D0 remain unchanged. The digits of hh are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D0=HEX hhhh - Load D0 with hex constant hhhh

opcode: 1Ahhhh
cycles: 6

Load the low-order four nibbles of D0 with the hex constant hhhh. The upper nibble of D0 remains unchanged. The digits of hhhh are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D0=HEX hhhhh - Load D0 with hex constant hhhhh

opcode: 1Bhhhhh
cycles: 7

Load all five nibbles of D0 with the hex constant hhhhh. The digits of hhhhh are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D1=(2) nn - Load 2 Nibbles Into D1

opcode: 1Dnn
cycles: 4

Load the low-order two nibbles of D1 with nn. The upper nibbles of D1 remain unchanged. Any overflow is ignored by the assembler. The assembled digits of nn are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D1=(4) nnnn - Load 4 Nibbles Into D1

opcode: 1Ennnn
cycles: 6

Load the low-order four nibbles of D1 with nnnn. The upper nibble of D1 remains unchanged. Any overflow is ignored by the assembler. The assembled digits of nnnn are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D1=(5) nnnnn - Load 5 Nibbles Into D1

opcode: 1Fnnnnn
cycles: 7

Load all five nibbles of D1 with nnnnn. Any overflow is ignored by the assembler. The assembled digits of nnnnn are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D1=A - Copy A to D1 (nibs 0-4)

opcode: 131
cycles: 8

The A field of register A is copied into Data pointer register D1. Carry is not affected.

D1=AS - Copy A to D1 short (nibs 0-3)

opcode: 139
cycles: 7

The lower 4 nibbles of A are copied into the lower 4 nibbles of Data pointer register D1. Carry is not affected.

D1=C - Copy C to D1 (nibs 0-4)

opcode: 135
cycles: 8

The A field of register C is copied into Data pointer register D1. Carry is not affected.

D1=CS - Copy C to D1 short (nibs 0-3)

opcode: 13D
cycles: 7

The lower 4 nibbles of C are copied into the lower 4 nibbles of Data pointer register D1. Carry is not affected.

D1=D1+ n - Add n to D1 (1<=n<=16)

opcode: 17x (x=n-1)
cycles: 7

Increment D1 by n. This instruction is always executed in HEX mode. Adjusts Carry.

D1=D1- n - Subtract n from D1 (1<=n<=16)

opcode: 1Cx (x=n-1)
cycles: 7

Decrement D1 by n. This instruction is always executed in HEX mode. Adjusts Carry.

D1=HEX hh - Load D1 with hex constant hh

opcode: 1Dhh
cycles: 4

Load the low-order two nibbles of D1 with the hex constant hh. The upper nibbles of D1 remain unchanged. The digits of hh are stored in the opcode in reverse order so that when the instruction is

executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D1=HEX hhhh - Load D1 with hex constant hhhh

opcode: 1Ehhhh
cycles: 6

Load the low-order four nibbles of D1 with the hex constant hhhh. The upper nibble of D1 remains unchanged. The digits of hhhh are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D1=HEX hhhhh - Load D1 with hex constant hhhhh

opcode: 1Fhhhhh
cycles: 7

Load all five nibbles of D1 with the hex constant hhhhh. The digits of hhhhh are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

D=-D fs - Two's complement of D into D

fs = A — opcode: FB, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: BbB, cycles: 3 + d

Complement the specified fs field of D. Complement is two's

complement if in HEX mode, ten's complement if in DEC mode. Carry is set if the field is not zero, else Carry is cleared.

D=-D-1 fs - One's complement of D into D

fs = A — opcode: FF, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: BbF, cycles: 3 + d

Perform a one's complement on the specified fs field of D. Carry is always cleared.

D=0 fs - Set D equal to 0

fs = A — opcode: D3, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Ab3, cycles: 3 + d

Set the specified fs field of D to zero. Carry is not affected.

D=C fs - Copy C to D

fs = A — opcode: D7, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Ab7, cycles: 3 + d

Copy the fs field of register C into the corresponding field of register D. Carry is not affected.

D=C-D fs - C minus D into D

fs = A	opcode: EF cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: BaF cycles: 3 + d

Set the specified fs field of register D to the inverse difference between itself and the corresponding field of register C. Adjusts Carry.

D=D!C fs - D OR C into D

fs = A	opcode: 0EFB cycles: 4 + d
fs = (P,WP,XS,X,S,M,B,W)	opcode: 0EaB cycles: 4 + d

Set the fs field of register D to its logical OR with the corresponding field of register C. Carry is not affected.

D=D&C fs - D AND C into D

fs = A	opcode: 0EF3 cycles: 4 + d
fs = (P,WP,XS,X,S,M,B,W)	opcode: 0Ea3 cycles: 4 + d

Set the fs field of register D to its logical AND with the corresponding field of register C. Carry is not affected.

D=D+1 fs - Increment D

---

fs = A	opcode: E7 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba7 cycles: 3 + d

Increment the specified fs field of register D by one. Adjusts Carry.

D=D+C fs - Sum of D and C into D

---

fs = A	opcode: C3 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Aa3 cycles: 3 + d

Set the specified fs field of register D to the sum of itself and the corresponding field of register C. Adjusts Carry.

D=D+D fs - Sum of D and D into D

---

fs = A	opcode: C7 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Aa7 cycles: 3 + d

Double the specified fs field of register D. Adjusts Carry.

D=D-1 fs - Decrement D

fs = A	opcode: CF cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AaF cycles: 3 + d

Decrement the specified fs field of register D by one. Adjusts Carry.

D=D-C fs - D minus C into D

fs = A	opcode: E3 cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: Ba3 cycles: 3 + d

Set the specified fs field of register D to the difference between itself and the corresponding field of register C. Adjusts Carry.

DATO=A fsd - Store into memory from A

fs = A	opcode: 140 cycles: 17
fs = B	opcode: 148 cycles: 14
fs = (P,WP,XS,X,S,M,W)	opcode: 150a cycles: 16 + d
fs = d	opcode: 158x (x=d-1) cycles: 15 + d

The amount of data (d nibbles) specified by fsd will be written to the memory address pointed to by D0 from the specified field of register A. The lowest-order nibble of the register field will be written to the lowest-addressed nibble of memory, proceeding toward the higher-order nibbles. If fs = d, d nibbles are written to

memory starting from nibble 0 of the register. See the section on "Storing Data Into Memory" earlier in this chapter.

DAT0=C fsd - Store into memory from C

fs = A	opcode: 144 cycles: 17
fs = B	opcode: 14C cycles: 14
fs = (P,WP,XS,X,S,M,W)	opcode: 154a cycles: 16 + d
fs = d	opcode: 15Cx (x=d-1) cycles: 15 + d

The amount of data (d nibbles) specified by fsd will be written to the memory address pointed to by D0 from the specified field of register C. The lowest-order nibble of the register field will be written to the lowest-addressed nibble of memory, proceeding toward the higher-order nibbles. If fs = d, d nibbles are written to memory starting from nibble 0 of the register. See the section on "Storing Data Into Memory" earlier in this chapter.

DAT1=A fs - Store into memory from A

fs = A	opcode: 141 cycles: 17
fs = B	opcode: 149 cycles: 14
fs = (P,WP,XS,X,S,M,W)	opcode: 151a cycles: 16 + d
fs = d	opcode: 159x (x=d-1) cycles: 15 + d

The amount of data (d nibbles) specified by fsd will be written to the memory address pointed to by D1 from the specified field of

register A. The lowest-order nibble of the register field will be written to the lowest-addressed nibble of memory, proceeding toward the higher-order nibbles. If fs = d, d nibbles are written to memory starting from nibble 0 of the register. See the section on "Storing Data Into Memory" earlier in this chapter.

DAT1=C fsd - Store into memory from C

fs = A	opcode: 145 cycles: 17
fs = B	opcode: 14D cycles: 14
fs = (P,WP,XS,X,S,M,W)	opcode: 155a cycles: 16 + d
fs = d	opcode: 15Dx (x=d-1) cycles: 15 + d

The amount of data (d nibbles) specified by fsd will be written to the memory address pointed to by D1 from the specified field of register C. The lowest-order nibble of the register field will be written to the lowest-addressed nibble of memory, proceeding toward the higher-order nibbles. If fs = d, d nibbles are written to memory starting from nibble 0 of the register. See the section on "Storing Data Into Memory" earlier in this chapter.

DCEX fs - Exchange Registers D and C

fs = A	opcode: DF cycles: 7
fs = (P,WP,XS,X,S,M,B,W)	opcode: AbF cycles: 3 + d

Exchange the fs fields of registers of D and C. Carry is not affected.

DSL fs - D Shift Left

fs = A — opcode: F3, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Bb3, cycles: 3 + d

Shift the contents of the specified fs field of register D left one nibble, without affecting the rest of the register. The nibble shifted off the left end of the field is lost. The new low-order nibble of the field is zero. The Sticky Bit (SB) is not affected.

DSLC - D Shift Left Circular

opcode: 813, cycles: 21

Circular shift register D left one nibble. Operates on all 16 digits. The Sticky Bit (SB) is not affected.

DSR fs - D Shift Right

fs = A — opcode: F7, cycles: 7

fs = (P,WP,XS,X,S,M,B,W) — opcode: Bb7, cycles: 3 + d

Shift the contents of the specified fs field of register D right one nibble, without affecting the rest of the register. The nibble shifted off the right end of the field is lost, but the Sticky Bit (SB) is set if the nibble was non-zero. The new high-order nibble of the field is zero.

DSRB - D Shift Right Bit

opcode: 81F
cycles: 20

Shift register D right one bit. Operates on all 16 digits. The bit shifted off the end is lost, but the Sticky Bit (SB) is set if it was non-zero. The new high-order bit of the register is zero.

DSRC - D Shift Right Circular

opcode: 817
cycles: 21

Circular shift register D right one nibble. Operates on all 16 digits. The Sticky Bit (SB) is set if the nibble shifted from low-order around to high-order position was non-zero.

GOC label - Go relative on carry

opcode: 4aa
cycles: 10 (GO)
3 (NO)

Short relative jump to label if Carry is set. label must be in the range:

addr - 128 <= label <= addr + 127

where addr is the address of the second nibble of the opcode. The address offset aa is in two's complement form and is relative to addr.

GOLONG label - Go Long

opcode: 8Caaaa
cycles: 14

Long relative jump to label unconditionally. label must be in the range:

addr - 32768 <= label <= addr + 32767

where addr is the address of the third nibble of the opcode. The address offset aaaa is in two's complement form and is relative to addr.

GONC label - Go relative on no carry

opcode: 5aa
cycles: 10 (GO)
3 (NO)

Short relative jump to label if Carry is clear. label must be in the range:

addr - 128 <= label <= addr + 127

where addr is the address of the second nibble of the opcode. The address offset aa is in two's complement form and is relative to addr.

GOSBVL label - Gosub very long to label

opcode: 8Faaaaa
cycles: 15

Absolute subroutine jump to aaaaa, which is the absolute address of label. See the GOSUB mnemonic.

GOSUB label - Gosub to label

opcode: 7aaa
cycles: 12

Relative subroutine jump to label. label must be in the range:

addr - 2048 <= label <= addr + 2047

where addr is the starting address of the next instruction. The address offset aaa is in two's complement form and is relative to addr.

As with all subroutine jumps, the address (addr) of the instruction following the gosub opcode is pushed onto the hardware return stack, so that when a corresponding return is executed, control resumes with the instruction at address addr.

As the return address is pushed onto the return stack, the bottom-most address on the stack is discarded. Therefore, the return stack always contains 8 addresses, and if pushes exceed pops by 8 levels, the bottom-most return addresses are lost. Since the interrupt system requires one level to process interrupts, only 7 levels of the return stack can be used by code which must execute when interrupts are enabled. See the RTN mnemonic for further information.

GOSUBL label - Gosub long to label

opcode: 8Eaaaa
cycles: 15

Long relative subroutine jump to label. label must be in the range:

addr - 32768 <= label <= addr + 32767

where addr is the starting address of the next instruction. The address offset aaaa is in two's complement form and is relative to addr. See the GOSUB mnemonic.

GOTO label - Jump relative

opcode: 6aaa
cycles: 11

Relative jump to label unconditionally. label must be in the range:

addr - 2048 <= label <= addr + 2047

where addr is the address of the second nibble of the opcode. The address offset aaa is in two's complement form and is relative to addr.

GOVLNG label - Jump very long

opcode: 8Daaaaa
cycles: 14

Unconditional jump to aaaaa, which is the absolute address of label.

GOYES label - Jump if Test is True

opcode: yy
cycles: included in the accompaning Test mnemonic cycle time.

GOYES is a mnemonic to specify part of a CPU test opcode. GOYES must always follow a test mnemonic. If the condition of the test is met, a jump is performed to label with Carry set. label must be in the range

addr - 128 <= label <= addr + 127

where addr is the starting address of the jump offset yy. If the test condition is not met, Carry is cleared and control passes to the next instruction. Compare with RTNYES.

INTOFF - Interrupt Off

opcode: 808F
cycles: 5

Disable the keyboard interrupt system. However, INTOFF does not disable the "ON" key or "INT" line interrupts. See the "HP-71 Hardware Specification" for more information.

INTON - Interrupt On

opcode: 8080
cycles: 5

Enable the keyboard interrupt system. See the "HP-71 Hardware Specification" for more information.

LC(m) n..n - Load C with constant (1<=m<=6)

opcode: 3xn..n (x=m-1)
cycles: 3+m

Load m digits of the expression n..n to the C register beginning at the P pointer position, and proceeding toward higher-order nibbles, with the ability to wrap around the register. See the section on "Loading Data From Memory" earlier in this chapter.

LCASC \A..A\ - Load C with ASCII constant

opcode: 3mc..c
(m = 2*(# of chars)-1;
c..c = ASCII codes)
cycles: 3+2*(# of chars)

Load up to 8 ASCII characters to the C register beginning at the P

pointer position, and proceeding toward higher-order nibbles, with the ability to wrap around the register. Each A represents an ASCII character. The ASCII characters are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

LCHEX h..h - Load C with hex constant

opcode: 3nh..h (n=# of digits-1)
cycles: 4+n

Load up to 16 hex digits into the C register beginning at the P pointer position, and proceeding toward higher-order nibbles, with the ability to wrap around the register. The hex digits are stored in the opcode in reverse order so that when the instruction is executed the data will be loaded into the register with the intended orientation. See the section on "Loading Data From Memory" earlier in this chapter.

MP=0 - Clear Module Pulled bit (MP)

opcode: 828
cycles: 3

Clears the Module Pulled bit (MP) and pulls the Module Pulled Interrupt line low. See CLRHST mnemonic.

NOP3 - Three nibble No-op

opcode: 420
cycles: 10 (GO/RTNYES)
3 (NO)

This mnemonic generates a GOC to the next instruction, effectively skipping three nibbles.

NOP4 - Four nibble No-op

opcode: 6300
cycles: 11

This mnemonic generates a GOTO to the next instruction, effectively skipping four nibbles.

NOP5 - Five nibble No-op

opcode: 64000
cycles: 11

This mnemonic generates a relative GOTO to +4 nibbles. The fifth nibble in the opcode is a place holder and is jumped over. The mnemonic effectively skips five nibbles.

OUT=C - Load 3 nibbles of OR

opcode: 801
cycles: 6

All nibbles of the Output register are loaded with the low-order three nibbles of C (X field).

OUT=CS - Load 1 nibble of OR

opcode: 800
cycles: 4

The least significant nibble of the Output register is loaded with the least significant nibble of the C register.

P=C n - Copy P pointer from C at Nibble n

opcode: 80Dn
cycles: 6

Copy nibble n of register C into the P pointer.

P=P+1 - Increment P Pointer

opcode: 0C
cycles: 3

Increment the P pointer. If P is incremented past F it will automatically wrap around to 0. This instruction is always executed in HEX mode. Adjusts carry.

P=P-1 - Decrement P Pointer

opcode: 0D
cycles: 3

Decrement the P pointer. If P is decremented past 0 it automatically wraps around to F. This instruction is always executed in HEX mode. Adjusts Carry.

P=n - Set P Pointer to n

opcode: 2n
cycles: 2

Set the P pointer to n.

R0=A - Copy A to register R0

opcode: 100
cycles: 19

The contents of the working register A is copied to the scratch register R0.

R0=C - Copy C to register R0

opcode: 108
cycles: 19

The contents of the working register C is copied to the scratch register R0.

R1=A - Copy A to register R1

opcode: 101
cycles: 19

The contents of the working register A is copied to the scratch register R1.

R1=C - Copy C to register R1

opcode: 109
cycles: 19

The contents of the working register C is copied to the scratch register R1.

R2=A - Copy A to register R2

opcode: 102
cycles: 19

The contents of the working register A is copied to the scratch register R2.

R2=C - Copy C to register R2

opcode: 10A
cycles: 19

The contents of the working register C is copied to the scratch register R2.

R3=A - Copy A to register R3

opcode: 103
cycles: 19

The contents of the working register A is copied to the scratch register R3.

R3=C - Copy C to register R3

opcode: 10B
cycles: 19

The contents of the working register C is copied to the scratch register R3.

R4=A - Copy A to register R4

opcode: 104
cycles: 19

The contents of the working register A is copied to the scratch register R4.

R4=C - Copy C to register R4

opcode: 10C
cycles: 19

The contents of the working register C is copied to the scratch register R4.

RESET - System reset

opcode: 80A
cycles: 6

The System Reset Bus Command is issued with all chips performing a local reset. The reset function will vary according to the chip type. See the "HP-71 Hardware Specification" for more information.

RSTK=C - Push C to Return Stack

opcode: 06
cycles: 8

Push the low-order 5 nibbles (A field) of the C register onto the Return Stack. The C register remains unchanged. See the GOSUB mnemonic.

RTI - Return from interrupt

opcode: 0F
cycles: 9

Return and re-enable the interrupt system. See the RTN mnemonic, and also the "HP-71 Hardware Specification."

RTN - Return

opcode: 01
cycles: 9

Return control to the top address on the hardware return stack. The top address on the hardware return stack is popped off and placed in the program counter PC. As the address is popped off the stack, a zero address is inserted at the bottom of the stack.

Therefore the hardware return stack always contains 8 addresses, and if more pops (returns) than pushes (gosubs) are performed, zeros will be read off the stack. Such an attempt to "return" to address 0 results in a memory reset, since the memory reset code of the operating system resides at address 0.

RTNC -Return on carry
---

opcode: 400
cycles: 10 (RTN)
3 (NO)

Return if Carry is set. See RTN mnemonic.

RTNCC - Return, clear carry
---

opcode: 03
cycles: 9

Return and clear Carry. See RTN mnemonic.

RTNNC - Return on no carry
---

opcode: 500
cycles: 10 (RTN)
3 (NO)

Return if Carry is not set. See RTN mnemonic.

RTNSC - Return, set carry
---

opcode: 02
cycles: 9

Return and set Carry. See RTN mnemonic.

RTNSXM - Return, set External Module Missing bit (XM)

opcode: 00
cycles: 9

Return and set the External Module Missing bit (XM). Since the opcode is zero, this mnemonic is executed on a jump to a non-existent memory device. See the "HP-71 Hardware Specification" for more information. See also the RTN mnemonic.

RTNYES - Return if Test is True

opcode: 00
cycles: included in the accompaning mnemonic cycle time.

RTNYES is a mnemonic to specify part of a CPU test opcode. RTNYES must always follow a test mnemonic. If the test condition is met, Carry is set and a return is executed. If the test condition is not met, control passes to the instruction following the RTNYES. Compare with the RTN and GOYES mnemonics.

SB=0 - Clear Sticky Bit (SB)

opcode: 822
cycles: 3

Clear the Sticky Bit (SB). See CLRHST mnemonic.

SETDEC - Set decimal

opcode: 05
cycles: 3

Set CPU arithmetic mode to decimal.

SETHEX - Set hexadecimal mode

opcode: 04
cycles: 3

Set CPU arithmetic mode to hexadecimal.

SHUTDN - System Shutdown

opcode: 807
cycles: 5

When this mnemonic is executed the CPU sends out the Shutdown Bus Command and stops its clock. Issuing the SHUTDN command with the least significant bit of the output register equal to 0 (i.e., if bit0=0) will immediately cause a cold start -- the PC is set to 0 and the CPU is not halted. This action is to insure that the ON key will be able to wake up the CPU. See the "HP-71 Hardware Specification" for more information.

SR=0 - Clear Service Request bit (SR)

opcode: 824
cycles: 3

Clear the Service Request bit (SR). See the CLRHST mnemonic.

SREQ? - Service Request
---

opcode: 80E
cycles: 7

This mnemonic sets the Service Request bit (SR) if any chip on the system bus requests service. When it is executed, a Service Request Bus Command is issued on the system bus to poll all chips for a Service Request. If any chip requests service, a bus line will be pulled high during the next strobe following the Service Request Bus Command. This value of the bus will be latched into the least significant nibble of the C register. The bus line pulled high determines the device type according to the following table.

Bit	Device
3	Unused
2	Card Reader
1	HP-IL Mailbox
0	Display Driver (timer)

If any bus line is high, the Service Request bit (SR) will be set. See the "HP-71 Hardware Specification" for more information. See also the ?SREQ and SR=0 mnemonics.

ST=0 n - Clear Program Status bit n
---

opcode: 84n
cycles: 4

Clear the Program Status bit selected by n.

ST=1 n - Set Program Status bit n
---

opcode: 85n
cycles: 4

Set the Program Status bit selected by n.

ST=C - C to Status

opcode: 0A
cycles: 6

Copy the low-order 12 bits of the C register (X field) into the low-order 12 bits of the Status register.

UNCNFG - Unconfigure

opcode: 804
cycles: 12

Load the low-order 5 nibbles (A field) of the C register into the data address register of each peripheral chip, with the device addressed by the C register unconfiguring. See the "HP-71 Hardware Specification" for more information.

XM=0 - Clear External Module Missing bit (XM)

opcode: 821
cycles: 3

Clear the External Module Missing bit (XM). This hardware status bit is set by the RTNSXM mnemonic. See the CLRHST mnemonic.