# SECTION 20

# Macro-Instructions

## 20.1 INTRODUCTION

Probably the most confusing thing to new users is determining how the macro-instruction set differs from the "core" Lisp routines (that is, those Lisp routines needed to define a minimum Lisp kernel). This instruction set (the macro-instructions) is a "pseudo" subset of the Lisp instruction set. In other words, these instructions are those that are implemented by microcode and are at the level below the compiler. It is the set available to the compiler in constructing functional content. Some Lisp instructions map directly into macro-instructions (therefore, you can get an ample description of what the instruction does by consulting the Explorer Lisp Reference manual), while others are a bit more complex and require more effort from the compiler. Still other macro-instructions don't correspond to basic Lisp functions at all, but are used in other aspects of the virtual machine implementation (such as in virtual memory or storage management). There are also some instructions here that are only available to the compiler and so are not classified as Lisp instructions (for example, Exchange).

The purpose of this section is to describe in more detail the macro-instruction set so that if you were the compiler or debugging compiler-generated code you could easily determine exactly what arguments are needed for a macro-instruction, what side effects a macro-instruction may have, and finally, what should be returned to you on the stack. Therefore, you will find in this section more detail on datatypes, exact error checking being performed, stack manipulation, and some history behind why some instructions exist and how they are used.

NOTE

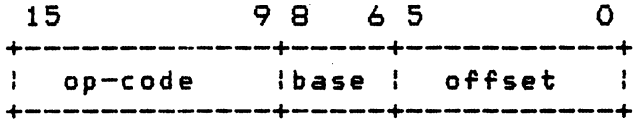
The section entitled The Disassembler in the Explorer Lisp Reference manual explains how to use the DISASSEMBLE function, how the macro-instruction set works and how to understand the behavior of code written in this instruction set. That section should be read before proceeding with this section.

NOTE

The instruction set for Release 3 is defined by the file SYS:UCODE;DEFOP which uses several macros that are defined in SYS:COMPILER;TARGET. Byte specifiers for the instruction fields can be found in SYS:UCODE;DEF-ELROY.

## 20.2 MAIN OPS

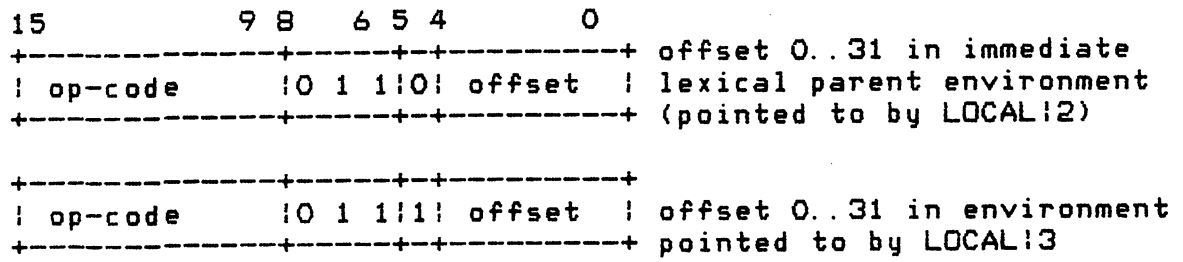
MAIN OP instructions (see Figure 20-1) have an operand address as part of the instruction (called base-arguments in this document) and may take additional operands from the stack. The disposition of the result is implied by the operation. All the base registers are 0-origin; that is, offset 0 is the first element.



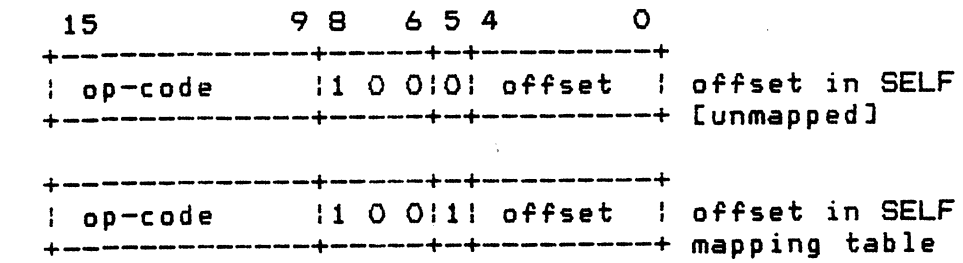

base register: 0 = FEF
1 = FEF+64
2 = FEF+128
3 = higher lexical context
4 = SELF mapping table
5 = local variables
6 = arguments
7 = PDL-POP [offset not used]

Figure 20-1 MAIN OP Instruction Format

For a base register value of 3, the offset field is interpreted as follows:



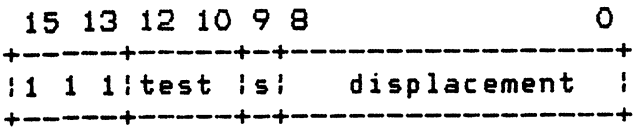

For a base register value of 4, the offset field is interpreted as follows:




MAIN-OPS are defined by the special form DEFOP.

## 20.3 SHORT BRANCHES

The Short Branch instruction format is shown in Figure 20-2. The op-code identifies the instruction as a branch and specifies the condition to be tested. The lower 9 bits of the instruction are a signed displacement relative to the address of the next instruction.




test: Test Condition
- 0 = NULL [else pop]
- 1 = NULL
- 2 = ATOM
- 3 = ZEROP
- 4 = SYMBOLP
- 5 = [unused]
- 6 = NULL [likely]
- 7 = unconditional

s: Sense
- 0 = branch when true
- 1 = branch when false

Figure 20-2 Short Branch Instruction Format

If the displacement cannot be encoded within 9 bits, then a Long Branch is used. Long branches are separate op-codes in the AUX-OP group (see below).

Short branch instructions are defined using the special form DEF-BRANCH-OP.

## 20.4 IMMEDIATE OPERATIONS

Immediate operations (see Figure 20-3) use the lower 9 bits of the instruction word in special ways. PUSH-NUMBER uses it as a FIXNUM operand. Various op-IMMED instructions use it as a signed integer operand. These are defined by using DEFOP with the :NO-REG option of Immed.

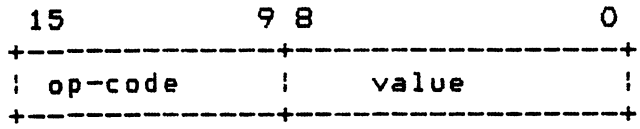
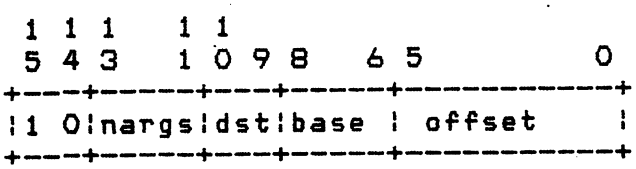



Figure 20-3 Immediate Operation Instruction Format

## 20.5 CALL INSTRUCTIONS

The instructions CALL-0 through CALL-6 have the format shown in Figure 20-4.




nargs: Number of Arguments

dst: Destination
0 = set indicators only
1 = push result on stack
2 = return from current frame
3 = replace current frame

Figure 20-4 Call Instruction Format

The base and offset fields are the same as for the MAIN-OPS and specify the function to be called. The function arguments are pushed on the stack before executing this instruction. If the number-of-arguments field contains a 7, then this is a CALL-N instruction, and the number of arguments is the last thing that was pushed on the stack.

The various call instructions are defined using the special form DEF-CALLOP.

## 20.6 MISC-OPS

MISC-OPS (see Figure 20-5) take their arguments from the stack and produce a result value which sets the indicators and is optionally pushed on the stack.

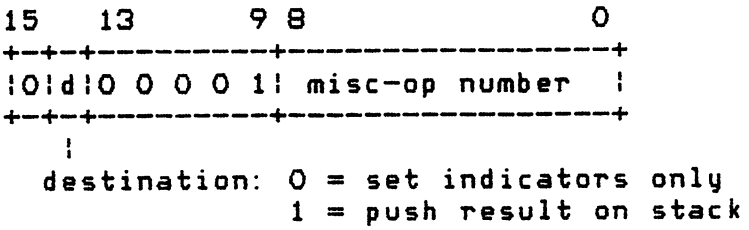



Figure 20-5 MISC-OP Instruction Format

The lower 9 bits of the instruction specify which of the many MISC-OPS is to be performed.

MISC-OPS are defined by using the special form DEF-MISC-OP.

## 20.7 AUX-OPS

AUX-OPS (see Figure 20-6) are similar to MISC-OPS except that they do not produce any result value.

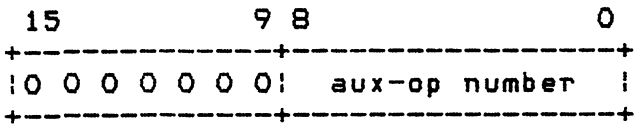
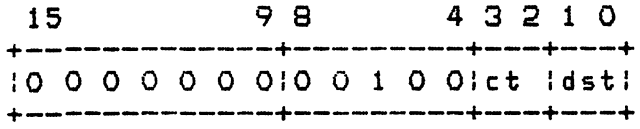



Figure 20-6 AUX-OP Instruction Format

The lower 9 bits of the instruction specify which of the AUX-OPS is to be performed.

AUX-OPS are defined using the special form DEF-AUX-OP. AUX-OPS can be considered to be divided into four groups: simple AUX-OPS, Complex Calls, Long Branches, and AUX-OPS with a count field.

### 20.7.1 AUX-OP Complex Call.

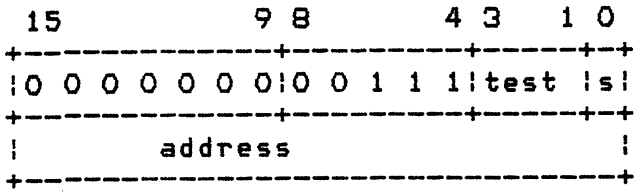



ct: Call Type
- 0 = Use call-info word. Push arguments, mapping table (optional), call-info word and function.
- 1 = APPLY (with one argument). Push argument and function to be called.
- 2 = [ unused ]
- 3 = LEXPR-FUNCALL-WITH-MAPPING-TABLE (one argument). Push argument, mapping table and function.

dst: Destination
- 0 = set indicators only
- 1 = push result on stack
- 2 = return from current frame
- 3 = replace current frame

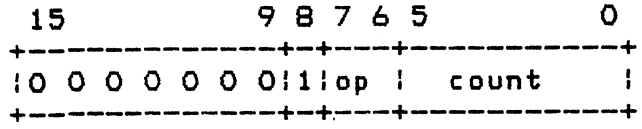
See the section on function calling for a description of the call-info word.

### 20.7.2 AUX-OP Long Branch.




The Test and Sense fields have the same values as for a short branch. Instead of a signed relative displacement, the second half-word of the instruction contains the new PC offset from the start of the FEF.

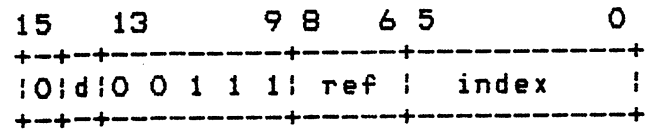
### 20.7.3 AUX-OPS With Count Field.




op: Operation
- 0 = unbind "count" special variables
- 1 = pop "count" values off stack
- 2 = return "count" values from stack
- 3 = [unused]

## 20.8 AREFI INSTRUCTIONS

This group of instructions (see Figure 20-7) is used for single-dimension array references having a small constant index. The index is an immediate value in the instruction, and the other operands are taken from the stack.



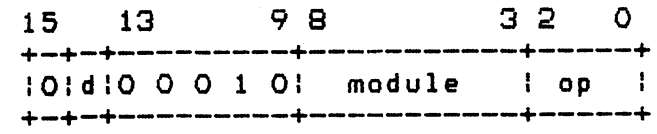

d: Destination
- 0 = indicators
- 1 = push

ref: Reference kind
- 0 = AREF [Zetalisp]
- 1 = ARRAY-LEADER
- 2 = %INSTANCE-REF
- 3 = AREF [Common Lisp]
- 4 = (SETF (AREF ...))
- 5 = (SETF (ARRAY-LEADER
- 6 = (SETF (%INSTANCE-REF
- 7 = [unused]

Figure 20-7 AREFI Instruction Format

## 20.9 MODULE GROUP

This group of instructions (see Figure 20-8) are similar to MISC-OPS, except that they tend to be specific for certain applications or environments, and the microcode that implements a module is not required to be present.




d: Destination
- 0 = set indicators only
- 1 = push result on stack

Figure 20-8 Module Group Instruction Format

The operation to be performed is specified by a 6-bit module number and a 3-bit operation code within that module. The currently assigned module numbers are:

0 TV (various %DRAW-<thing> functions)
1 Mouse

with more to be defined later.

## 20.10 UCODE ENTRIES

A Ucode entry (which stands for microcode entry) is not really a macro-instruction at all. Rather it is a functional object; a word of data type DTP-U-ENTRY. It can be funcalled, applied to arguments, passed as a value, and so forth just like any other Lisp functional object. The use of the pointer field of a U-ENTRY is detailed in the section on Internal Storage Formats.

Ucode entries are listed here because they closely resemble MISC-OPs and for completeness because all such functions must be defined by the DEFOP file. A Ucode entry, like a MISC-OP, is a microcoded function which takes its arguments from the stack. However, the number of arguments is variable, so the function can have an &REST arg.

## 20.11 MACROCODE INSTRUCTION SET

The following is the syntax line for a macro-instruction:

SOME-MACROINSTRUCTION arg1 arg2 [opcode] FORMAT Level description

- The name of the macro-instruction will be in upper case.

- Arguments are listed; they are to be pushed on the stack, arg1 being the first one pushed on the stack. The last argument listed is the one on top of the stack. If an argument is described with the prefix immed-, then the argument is not pushed on the stack, rather it is included in the instruction itself (see the IMMEDIATE instruction description above). If an argument is described with the prefix base- then the argument is also not pushed on the stack but rather addressed with the MAIN-OP base-offset scheme described above.

- Brackets ([]) are used to enclose the opcode for the macro-instruction (specified in octal).

- The FORMAT of the macro-instruction will be on the far right and in upper case. It will be MAIN-OP, SHORT-BRANCH, IMMEDIATE, CALL, MISC-OP, AUX-OP, AREFI, or MODULE(MODULE-NAME).

- Level is either Lisp-Function or blank. This indicates if there is a corresponding Lisp function for this macro-instruction by the same name. If there is a corresponding Lisp function for this macro-instruction by another name, then the Lisp function will be referred to in description.

- A brief description will then follow.

- All MISC-OP instructions pop arguments that are on the stack and return a value on the top of stack unless otherwise indicated.

- All MAIN-OP instructions use the base-offset scheme to address their first argument (base-argument) and all other arguments are popped off the stack.

- Most instructions set the indicators. Some exceptions are: branches, (AUX) POP-PDL, (AUX)

UNBIND, LEXICAL-UNSHARE, and (AUX) LEXICAL-UNSHARE-ALL.

The following is an alphabetic list of macro-instructions.

ABS num [513] MISC-OP Lisp-Functio
Returns the absolute value of num which can be any type of number.

ADD-IMMED num, immed-y [43] IMMEDIATE
Returns the sum of num and the immediate operand, immed-y, which is part of the ADD-IMMED instruction.

ALOC array, &REST subscripts [2] UCODE ENTRY
Returns a locative to the element of array specified by subscripts.

AP-1 array, index [645] MISC-OP Lisp-Functio
Returns a locative to the element of array specified by index. This is the one-dimensional case of ALOC.

AP-1-FORCE array,index [652] MISC-OP Lisp-Functio
Returns a locative to the element of array specified by index. Array is treated (forced) as one-dimensional. That is, it is indexed with a single subscript regardless of its rank.

AP-2 array, sub1, sub2 [646] MISC-OP Lisp-Functio
Returns a locative to the element of array specified by subscripts sub1 and sub2.

AP-3 array, sub1, sub2, sub3 [647] MISC-OP Lisp-Functio
Returns a locative to the element of array specified by subscripts sub1, sub2 and sub3.

AP-LEADER array, index [644] MISC-OP Lisp-Functio
Returns a locative to the leader element of array specified by the subscript index.

APPLY-TO-INDS fn, args [104] AUX-OP
Apply function fn to the list of arguments args. Set the indicators with the value returned.

APPLY-TO-PUSH fn, args [105] AUX-OP
Apply function fn to the list of arguments args. Push the returned value on the stack.

APPLY-TO-RETURN fn, args [106] AUX-OP
Apply function fn to the list of arguments args. Return the result value from the current function.

APPLY-TO-TAIL-REC fn, args [107] AUX-OP
Apply function fn to the list of arguments args.

Replace the current stack frame and return the result value.

ZLC:AR-1 array, index [641] MISC-OP Lisp-Function
Returns the element of the one-dimensional array array specified by index. Array must be a one-dimensional array and index must be a FIXNUM. If index is less than zero or greater than the largest index permissible, then a SUBSCRIPT-OOB error is signalled. If array is not one-dimensional, then ARRAY-NUMBER-DIMENSIONS is signalled. The type of result depends on the type of array.

AR-1-FORCE array, index [651] MISC-OP Lisp-Function
Returns the element of the array array specified by index. Array is treated (forced) as a one-dimensional array; i.e., it is indexed with a single subscript regardless of its actual rank.

AR-2 array, sub1, sub2 [642] MISC-OP Lisp-Function
Exactly like AR-1 except array must be two-dimensional.

ZLC:AR-2-REVERSE array, sub2, sub1 [650] MISC-OP Lisp-Function
Returns the element of the two-dimensional array array. See the Explorer Lisp Reference manual for a discussion of this instruction.

AR-3 array, sub1, sub2, sub3 [643] MISC-OP Lisp-Function
Exactly like AR-1 except array must be three-dimensional.

AREF array, &REST subscripts [1] UCODE ENTRY
Returns the element of the array array specified by subscripts.

ARRAY-ACTIVE-LENGTH array [662] MISC-OP Lisp-Function
Returns the number of "active" elements in array. If the array has a fill pointer then the fill pointer value is returned, else the number of elements is returned.

ARRAY-DIMENSION array, dimension [665] MISC-OP Lisp-Function
Returns the length of dimension dimension of array. The first dimension is number 0.

ARRAY-HAS-FILL-POINTER-P array [241] MISC-OP Lisp-Function
Returns T if array has a leader and leader element 0 is a FIXNUM. Otherwise, returns the symbol NIL. If array is not an array an ARGTYP error is signalled.

ARRAY-HAS-LEADER-P array [234] MISC-OP Lisp-Function

Returns T if array has a leader. Otherwise, returns the symbol NIL. If array is not an array an ARGTYP error is signalled.

ARRAY-IN-BOUNDS-P array &REST subscripts [5] UCODE ENTRY
Returns T if the indices are in bounds for the dimensions of array. Otherwise, returns the symbol NIL.

ARRAY-LEADER array, index [640] MISC-OP Lisp-Function
Returns the array leader element of array specified by index. If array is not an array an ARGTYP error is signalled. If array does not have a leader the ARRAY-HAS-NO-LEADER error is signalled. If index is not a FIXNUM the ARGTYP error is signalled. Finally, if index is greater than or equal to the length of the leader the SUBSCRIPT-OOB error is signalled.

ARRAY-LEADER-LENGTH array [663] MISC-OP Lisp-Function
Returns the length of the array leader of array. If array is not an array the ARGTYP error is signalled.

ARRAY-LENGTH array [660] MISC-OP Lisp-Function
Returns the length of array. Does not take into account the fill pointer. Compare this macro-instruction with ARRAY-ACTIVE-LENGTH.

ARRAY-PUSH array, value [340] MISC-OP Lisp-Function
Add value as an element at the end of array. The fill pointer (leader element 0) is the index of the next element to be added. Returns NIL and doesn't update the fill pointer if array is full, otherwise returns the index of the element added. Does not automatically increase the size of the array like ARRAY-PUSH-EXTEND.

ARRAY-RANK array [664] MISC-OP Lisp-Function
Returns the rank or number of dimensions of array.

ARRAYP base-arg1 [31] MAIN-OP Lisp-Function
arg1 [600] MISC-OP Lisp-Function
Returns T if base-arg1 (or arg1) is an array (has DTP-ARRAY datatype). Otherwise the symbol NIL is returned.

AS-1 value, array, index [321] MISC-OP Lisp-Function
Stores value into the one-dimensional array array specified by index. Array must be an array and index must be a FIXNUM, otherwise the ARGTYP error is signalled. If index is less than zero or greater than the largest index permissible, then a SUBSCRIPT-OOB error is signalled. If array

is not one-dimensional, then ARRAY-NUMBER-DIMENSIONS is signalled. Returns value.

AS-1-FORCE value, array, index [324] MISC-OP Lisp-Functi
Stores value into the array array specified by index. Array is treated (forced) as one-dimensional; i.e., it it indexed with the single subscript index regardless of its rank. Returns value.

AS-2 value, array, sub1, sub2 [322] MISC-OP Lisp-Functi
Exactly like AS-1 except array is two-dimensional.

AS-2-REVERSE value, array, sub1, sub2 [325] MISC-OP Lisp-Functi
Stores value into the two-dimensional array array. See the Explorer Lisp Reference manual for a discussion of this instruction.

AS-3 value, array, sub1, sub2, sub3 [323] MISC-OP Lisp-Functi
Exactly like AS-1 except array is three-dimensional.

ASET value, array, &REST subscripts [0] UCODE ENTRY
Stores value into the array array specified by subscripts.

ASH n, nbits [531] MISC-OP Lisp-Functi
Shift n arithmetically by nbits. N may be a DTP-FIXNUM or a DTP-EXTENDED-NUMBER.

ASSQ x, alist [610] MISC-OP Lisp-Functi
Search alist by comparing the CAR of each element for being EQ to x. Returns the CDR of the matching element if a match is found. Otherwise returns the symbol NIL.

ATOM x [564] MISC-OP Lisp-Functi
Returns the symbol NIL if x is a list. Otherwise, returns T.

BIGNUM-TO-ARRAY bignum, base [654] MISC-OP Lisp-Functi
Converts bignum into an array. Bignum is expressed in base and placed into an ART-Q array. The sign of bignum is ignored.

BIND loc, val [200] MISC-OP
Set the value of loc to val and save the old value of loc on the special binding stack.

BIND-CURRENT base-loc [154] MAIN-OP
Save the value of base-loc on the special binding stack. Any other stores into base-loc will not corrupt its previous value.

BIND-NIL base-loc [151] MAIN-OP
Set the value of base-loc to NIL and save the old value

on the special binding stack.

BIND-POP newval, base-loc [153] MAIN-OP
Set the value of base-loc to newval and save the old value on the special binding stack.

BIND-T base-loc [152] MAIN-OP
Set the value of base-loc to T and save the old value on the special binding stack.

BIT-VECTOR-P object [250] MISC-OP Lisp-Functic
Returns T if object is a bit vector, otherwise return NIL. A bit vector is defined to be an array of rank 1 whose elements are restricted to 0 and 1; i.e., a one-dimensional array of array type ART-1b.

BITBLT alu width from-array from-x from-y to-array to-x to-y [345] MISC-OP Lisp-Functic

BOUNDP symbol [240] MISC-OP Lisp-Functic
Returns T if the value cell of symbol is unbound. A cell is unbound if its data type is DTP-NULL.

BREAKPOINT [1] AUX-OP

C**R list [4,6] MISC-OP Lisp-Functio
Returns the C**R of list. Signals the ARGTYP error if list (or any required intermediary list) is not a list. The error check may be overridden if taking the CAR of a symbol or number is allowed. C**R may be CDAR or CAAR.

C***R list [11-17] MISC-OP Lisp-Functio
Returns the C***R of list. Signals the ARGTYP error if list (or any required intermediary list) is not a list. The error check may be overridden if taking the CAR of a symbol or number is allowed. C***R may be CAAAR, CAADR, CADAR, CDAAR, CDADR, CDDAR, or CDDDR.

C****R list [20-37] MISC-OP Lisp-Functio
Exactly like C***R. C****R may be CAAAAR, CAAADR, CAADAR, CAADDR, CADAAR, CADADR, CADDAR, CADDDR, CDAAAR, CDAADR, CDADAR, CDADDR, CDDAAR, CDDADR, CDDDAR, or CDDDDR.

CALL-* base-func [100, 104, 110, 114, 120, 124, 130] CALL-OP
Call the function base-func. * may be from 0 to 6 respectively, and represents the number of arguments for base-func that have already been pushed on the stack. Func may be any functional argument.

CALL-N n, base-func [134] CALL-OP
This is used to call functions with greater than 7 arguments.
Call the function base-func with n arguments.
The n arguments have already been pushed on the stack.
Base-func may be any functional argument.

CAR-SAFE object [620] MISC-OP Lisp-Function

CARCDR list [622] MISC-OP
Returns the cdr of list. Sets the indicators based
on CAR of list.

CDR-SAFE object [621] MISC-OP Lisp-Function

CEILING-1 dividend, divisor [551] MISC-OP
Returns dividend divided by divisor rounded up.
The remainder is not returned. To receive both
the quotient and the remainder see CEILING-2.

CEILING-2 dividend, divisor [555] MISC-OP
Returns dividend divided by divisor, rounded up,
and the remainder. Therefore, returns 2 values.

CHAR-INT character [704] MISC-OP Lisp-Function
Returns a FIXNUM whose value corresponds to
character.

CHARACTERP object [254] MISC-OP Lisp-Function
Returns T if object is a character, otherwise returns
NIL. Object is a character if it is defined with
a DTP-CHARACTER data type.

CLOSURE symbol-list, function [717] MISC-OP Lisp-Function
Returns a dynamic closure (DTP-CLOSURE),
closing function over the special variables
in symbol-list.

COMMON-AR-1 array, index [670] MISC-OP Lisp-Function
Returns the element of the one-dimensional array
array specified by index. Array must be a
one-dimensional array and index must be a FIXNUM.
If index is less than zero or greater than the
largest index permissible, then a SUBSCRIPT-OOB
error is signalled. If array is not
one-dimensional, then ARRAY-NUMBER-DIMENSIONS is signalled.
The type of result depends on the type of array.
This differs from AR-1 for array types string and fat-string.
AR-1 will return FIXNUMs for elements within these arrays
while the COMMON-LISP-AR-n instructions return characters.

COMMON-LISP-AR-1-FORCE array, index [673] MISC-OP Lisp-Function
Returns the element of the array array specified by

index. Array is treated (forced) as a one-dimensional array; i.e., it is indexed with a single subscript regardless of its actual rank. See COMMON-LISP-AR-1 for the difference between this and AR-1-FORCE.

COMMON-LISP-AR-2 array, sub1, sub2 [671] MISC-OP Lisp-Function
Exactly like COMMON-LISP-AR-1 except array must be two-dimensional.

COMMON-LISP-AR-3 array, sub1, sub2, sub3 [672] MISC-OP Lisp-Function
Exactly like COMMON-LISP-AR-1 except array must be three-dimensional.

COMMON-LISP-AREF array, &REST indices UCODE ENTRY
This is no longer a macro-instruction but rather a function.

COMMON-LISP-ELT sequence, index [626] MISC-OP Lisp-Function
Returns the element of sequence specified by index, which must be a positive FIXNUM or the ARGTYP error is signalled. Sequence may be a one-dimensional array (vector) or a list.

COMMON-LISP-LISTP object [243] MISC-OP Lisp-Function
The Common Lisp version of LISTP. Returns T if object is NIL or a cons, otherwise returns NIL. LISTP returns NIL if object is NIL.

COMPLEX-CALL callinfo, function [100] AUX-OP
Calls the function function with the supplied call-info word. If there are arguments for function then they are already on the stack. The complex call instructions basically enable the manipulation of the call-info word. Thus, they are used mainly to achieve Lexpr calls without providing the self-mapping table (there is another macro-instruction for Lexpr calls with self-mapping tables: LEXPR-FUNCALL-WITH-MAPPING-TABLE), and Multiple value return calls. See the section on function calling for more information.

COMPLEX-CALL-TO-* callinfo, function [100-103] AUX-OP
The four forms of COMPLEX-CALL.
Calls the function function with the supplied call-info word. If there are arguments for function then they are already on the stack.
* may be INDS, PUSH, RETURN, or TAIL-REC.
See the section on calling for information about the four call destinations.

COMPLEXP object [260] MISC-OP Lisp-Function
Returns T if object is a complex number, otherwise returns NIL.

CONS car, cdr [412] MISC-OP Lisp-Functic
Constructs a list cell (cons) that has a CAR of car and a CDR of cdr. The storage is allocated in the default consing area specified by the variable DEFAULT-CONS-AREA.

CONS-IN-AREA car, cdr, area [413] MISC-OP Lisp-Functic
Exactly like CONS except the storage is allocated in the argument area. Signals the ARGTYP error if area is not a FIXNUM or a symbol with a FIXNUM in its value cell.

CONSP-OR-POP object [304] MISC-OP

COPY-ARRAY-CONTENTS from, to [342] MISC-OP Lisp-Functic
Copy all the elements from the array FROM to the array TO. If TO is longer than FROM, TO is filled with zeros (if a numeric array) or NILs. If either array is multidimensional, its elements are used in the order they are stored in memory.

COPY-ARRAY-CONTENTS-AND-LEADER from to [343] MISC-OP Lisp-Functic
Exactly like COPY-ARRAY-CONTENTS except the leader slots are also copied.

COPY-ARRAY-PORTION from-array from-start from-end to-array to-start to-end
[344] MISC-OP Lisp-Functic
Exactly like COPY-ARRAY-CONTENTS except from-start and from-end are indices in from-array indicating the portion to copy. To-start and to-end are indices in to-array indicating where to place the elements.

DEPOSIT-FIELD value ppss fixnum [526] MISC-OP Lisp-Functic
Returns a number which in the byte ppss matches value and the rest matches fixnum. ppss is a field specifier as in LDB.

DISPATCH index base-disptable [72] MAIN-OP
Allows a multiway transfer of control. The address field is an immediate value used as a 9-bit offset from the beginning of the FEF; this is the beginning of a table of PC values. The table entry at the offset specified by the index argument becomes the new program counter value.

DPB value ppss num [524] MISC-OP Lisp-Functic
The inverse of LDB. The low order ss bits of value replace the field of the same size in num.

Never changes the sign of the quantity DPB'ed into. If pp is above the current size of num, the quantity is sign extended until it is long enough to accommodate the DPB. Returns either a FIXNUM or a BIGNUM.

Value and ppss must be FIXNUMs and ppss specify a field less than 25 bits. Num must be FIXNUM or a BIGNUM.

ENDP object [255] MISC-OP Lisp-Functio
Returns T if object is NIL. Returns NIL if object is a cons (DTP-LIST). Otherwise, signals the ARGTYP error.

EQ x, base-y [23] MAIN-OP Lisp-Functio
EQ x, y [574] MISC-OP
Returns T if x and base-y are the exact same Lisp object, otherwise returns NIL.

EQ-IMMED x, immed-y [3] IMMEDIATE
Exactly like EQ except immed-y is taken from the instruction itself.

EQ-T object [261] MISC-OP
Returns T if object is the Lisp object T, otherwise returns NIL.

EQL x, base-y [24] MAIN-OP Lisp-Functio
EQL x, y [573] MISC-OP Lisp-Functio
See the Explorer Lisp Reference manual.

EQUAL x, base-y [25] MAIN-OP Lisp-Functio
EQUAL x, y [562] MISC-OP
Returns T if x and base-y are EQ. If x and base-y are numbers then they are equal if they have the same value and type. Two conses are equal if the CARs are equal and their CDRs are equal. Two strings are equal if they have the same length, and the characters composing them are the same. All other objects are equal if and only if they are EQ.

EQUALP x, base-y [26] MAIN-OP Lisp-Functio
EQUALP x, y [575] MISC-OP
See the Explorer Lisp Reference manual.

EXCHANGE [10] AUX-OP
Swaps the top two items on the stack.
Accepts and returns no values.

EXPT base, exponent [536] MISC-OP Lisp-Functio
Returns base raised to the power exponent.

FBOUNDP symbol [237] MISC-OP lisp
Returns T if the function cell of symbol does not

contain the unbound marker, otherwise return NIL.
Symbol must be a symbol or the ARGTYP error is
signalled.

FIND-POSITION-IN-LIST element, list [616] MISC-OP Lisp-Function
Returns the numeric index in list at which element
is found (uses EQ), unlike MEMQ which may return
the rest of list beginning with element. Otherwise
returns NIL. The index returned is zero-based.
List must be a list or the ARGTYP error is signalled.

ZLC:FIX number [510] MISC-OP Lisp-Function
See the Explorer Lisp Reference manual.

FIXNUMP base-object [34] MAIN-OP Lisp-Function
object [602] MISC-OP Lisp-Function
Returns T if object is a FIXNUM, otherwise return NIL.

FIXP x [561] MISC-OP
Returns T if x is an integer, otherwise returns
NIL disregarding the datatype of x.

FLOAT-EXPONENT flonum [516] MISC-OP Lisp-Function
Returns the exponent of flonum as a FIXNUM.

FLOAT-FRACTION flonum [517] MISC-OP Lisp-Function
Returns flonum modified to contain 0 as its exponent.
The result is either zero or has absolute value at
least 1/2 and less than one.

FLOATP x [230] MISC-OP Lisp-Function
Returns T if x is a floating point number,
otherwise returns NIL.

FLOOR-1 dividend, divisor [550] MISC-OP
Returns dividend divided by divisor, rounded down.

FLOOR-2 dividend, divisor [551] MISC-OP
Returns dividend divided by divisor, rounded down,
and the remainder. Therefore, this instruction
returns 2 values.

FUNCTION-CELL-LOCATION symbol [633] MISC-OP Lisp-Function
Returns a locative to symbol's function cell.
Symbol must be a symbol or the ARGTYP error is
signalled.

GCD num1 num2 [543] MISC-OP
Returns the GCD (Greatest Common Divisor) of num1
and num2. Therefore, num1 or num2 will be returned.
Both arguments must be numbers or the ARGTYP error is
signalled.

G-L-P array [653] MISC-OP Lisp-Function
Returns a list with the contents of array. Array must be an array of type ART-Q-LIST or the ARGTYP error is signalled.

GET-LEXICAL-VALUE-CELL env-list, symbol-cell-location [722] MISC-OP Lisp-Function

GET-LOCATION-OR-NIL symbol property [72] MISC-OP Lisp-Function
Returns a locative to the plist location containing the value of property. Symbol can be a symbol, instance, or disembodied property list. If property is not found, returns NIL.

GETL symbol indicator-list [71] MISC-OP Lisp-Function
Find any of the properties in indicator-list, on symbol. Whichever of those properties occurs first in the property list is used. The value is a pointer to the cell in the property list that points to the indicator. The CADR of the value is the property's value.

HALT [3] AUX-OP
Halts the processor. This is used by hardware debugging to possibly evaluate the state of the machine. This should not normally be used to shutdown the system, rather %CRASH should be used.

HAULONG integer [515] MISC-OP Lisp-Function
Returns the number of bits in integer as a FIXNUM. For example, the size of #o777 is nine. Integer may be either a FIXNUM or a BIGNUM.

INT-CHAR fixnum [703] MISC-OP Lisp-Function
Returns a character whose value corresponds to fixnum.

INTEGERP base-object [35] MAIN-OP Lisp-Function
Returns T if base-object is an integer, otherwise returns NIL for other numbers and non-numbers.

INTERNAL-CHAR-EQUAL ch1 ch2 [232] MISC-OP
Compares two characters that are either FIXNUMs or DTP-CHARACTER objects. If the character codes are equal then T is returned. Otherwise, if ALPHABETIC-CASE-AFFECTS-STRING-COMPARISON is non-NIL, NIL is returned. If the characters are different and case doesn't matter, T is returned if one is the alphabetic uppercase of the other, otherwise NIL is returned. (NOTE: notice how those macro-instructions with the prefix INTERNAL- are not defined with the Lisp-Function level.)

INTERNAL-FLOAT number [512] MISC-OP
Returns number if number is already a floating-point. Otherwise, number is converted to a single-float number and returned. This is like ZLC:FLOAT.

INTERNAL-GET-2 symbol property [70] MISC-OP
Returns symbol's property property, otherwise NIL is returned. Returns NIL if symbol does not have a property list or is of a type that does not have properties. Symbol may be a symbol, list, locative, or an instance.

INTERNAL-GET-3 symbol property default [73] MISC-OP
Similar to INTERNAL-GET-2 except returns default instead of NIL if property is not found.

LAST list [611] MISC-OP Lisp-Functio
Returns the last cons cell of list. Signals the ARGTYP error if list is not of the type list.

LDB ppss, num [520] MISC-OP Lisp-Functio
Returns the ss number of bits starting at bit pp.

LDB-IMMED num, immed-ppss [44] IMMEDIATE
Exactly like LDB except immed-ppss is taken directly from the instruction opcode. Immed<8:4> is the 5 bit pp (position) and Immed<3:0> is the 4 bit ss (length).

LENGTH list-or-array [612] MISC-OP Lisp-Functio
Returns the number of elements in list-or-array if list-or-array is a list. Returns the array active length if list-or-array is an array. The array's active length is the value of its fill-pointer if it exists or the number of elements in the array.

LENGTH-GREATERP list-or-array value [231] MISC-OP Lisp-Functio
Tests whether list-or-array has more than the number of elements indicated by value without using LENGTH, and thus without going any farther down the list than necessary. Returns T if the number of elements is greater than value, otherwise returns NIL.

LEXICAL-UNSHARE [76] MAIN-OP
LEXICAL-UNSHARE-ALL [31] AUX-OP
See the Closures section.

LEXPR-FUNCALL-WITH-MAPPING-TABLE arglist, mapping-table, fn
[114] AUX-OP
Lexpr-funcalls the function fn with the supplied mapping-table and one argument, arglist.

See the section on function calling for more information concerning Lexpr-funcalls.

LEXPR-FUNCALL-WITH-MAPPING-TABLE-TO-*
arglist mapping-table fn [114-117] AUX-OP
The four forms of LEXPR-FUNCALL-WITH-MAPPING-TABLE. * may be INDS, PUSH, RETURN, or TAIL-REC, respectively. LEXPR-FUNCALL-WITH-MAPPING-TABLE-TO-INDS is the same as LEXPR-FUNCALL-WITH-MAPPING-TABLE.

LIST &REST elements [10] UCODE ENTRY
See the Explorer Lisp Reference manual.

LIST* first, &REST elements [11] UCODE ENTRY
See the Explorer Lisp Reference manual.

LIST*-IN-AREA area first &REST elements [13] UCODE ENTRY
See the Explorer Lisp Reference manual.

LIST-IN-AREA area, &REST elements [10] UCODE ENTRY
See the Explorer Lisp Reference manual.

LISTP x [32] MAIN-OP Lisp-Functio
LISTP x [576] MISC-OP
Returns T if x is a list or NIL, otherwise returns NIL. Note that this is the Common Lisp LISTP.

LOAD-FROM-HIGHER-CONTEXT contex-desc [720] MISC-OP
LOCATE-IN-HIGHER-CONTEXT contex-desc [721] MISC-OP
See Closures section.

LOCATE-IN-INSTANCE instance, symbol [710] MISC-OP Lisp-Functio
Returns a locative to the instance variable symbol of instance.

LOCATE-LEXICAL-ENVIRONMENT env-num [77] MAIN-OP
See Closures section.

LONG-BR [176] AUX-OP
Unconditional macrocode branch. Instead of a signed relative displacement like short-branches, the second halfword of the instruction contains the new PC offset relative to the start of the FEF (as opposed to relative from the current instruction).

LONG-BR-* [160-175] AUX-OP
Conditional macrocode branch. See LONG-BR for new PC generation scheme. * may be NULL-ELSE-POP, NOT-NULL-ELSE-POP, NULL, NOT-NULL, ATOM, NOT-ATOM, ZEROP, NOT-ZEROP, SYMBOLP, NOT-SYMBOLP, NULL-LIKELY or NOT-NULL-LIKELY.
See section on short branches for a description

of the different conditions.

LONG-PUSHJ [177] AUX-OP
Unconditional subroutine call. The destination of the call is computed like in LONG-BR. The macrocode subroutine will return POPJ.

LSH n, nbits [530] MISC-OP Lisp-Function
Return n logically shifted (zero fill) by nbits. The sign of nbits controls the direction of the shift. If nbits is negative, the shift is to the right. n and nbits must both be FIXNUMs or the ARGTYP error is signalled.

MAKE-EPHEMERAL-LEXICAL-CLOSURE envdesc, function [724] MISC-OP
MAKE-LEXICAL-CLOSURE envdesc, function [723] MISC-OP
See Closures section

MASK-FIELD ppss, fixnum [522] MISC-OP Lisp-Function
Returns fixnum with all but the ppss byte replaced with zeros.

MEMQ x, list [613] MISC-OP Lisp-Function
Returns the sublist of list beginning with the first occurrence of x (using EQ), otherwise return NIL.

MINUS number [514] MISC-OP Lisp-Function
Returns the negative of number. Number must be of type number of the ARGTYP error is signalled.

MINUSP base-x [37] MAIN-OP Lisp-Function
MINUSP number [567] MISC-OP
Returns T if base-x is negative (strictly less than zero), otherwise returns NIL. The ARGTYP error is signalled if number is not of type Number.

MOVEM base-loc [141] MAIN-OP
Copies the value at the top of stack to base-loc. Does not pop the top of stack nor does it return a value.

NAMED-STRUCTURE-P object [603] MISC-OP Lisp-Function
Returns the symbol name of object if it is a named-structure array, otherwise return NIL. See the section on internal storage for a description of named-structures and the location of their name. This instruction will also return NIL if the name of object is not a symbol or a closure.

NCONS car [410] MISC-OP Lisp-Function
Constructs a cons (list cell) that has a CAR of car and a CDR of NIL. The storage is allocated in the

area specified by the variable DEFAULT-CONS-AREA.

NCONS-IN-AREA car, area [411] MISC-OP Lisp-Function
Exactly like NCONS except the storage is allocated in area. area must be a FIXNUM or a symbol that evaluates to a FIXNUM or the ARGTYP error is signalled.

NLISTP x [235] MISC-OP Lisp-Function
Returns T if x is an atom, otherwise returns NIL. This instruction will return NIL for NIL.

NOT x [563] MISC-OP Lisp-Function
Returns T if x is NIL, otherwise returns NIL.

NOT-INDICATORS [77] MISC-OP
Returns T if and only if the indicators are null, otherwise returns NIL.

NSYMBOLP x [236] MISC-OP Lisp-Function
Returns T if x is not a symbol, otherwise returns NIL.

NTH n, list [614] MISC-OP Lisp-Function
Returns the nth element of list (0-origin).

NTHCDR n, list [615] MISC-OP Lisp-Function
Returns list with the first n elements discarded. This is the equivalent of performing the CDR n times.

NUMBERP x [30] MAIN-OP Lisp-Function
NUMBERP x [565] MISC-OP
Returns T if x is of type Number, otherwise returns NIL.

PDL-WORD n [40] MISC-OP
Returns the value on the PDL (cached stack) that is n below the current PDL-pointer. This assumes that the selected word is on the PDL (cached stack) and never fetches from memory.

PLUSP x [36] MAIN-OP Lisp-Function
PLUSP x [566] MISC-OP
Returns T if x is positive (strictly greater than zero), otherwise returns NIL. The ARGTYP error is signalled if x is not of type Number.

POP obj, base-loc [140] MAIN-OP
Removes (pops) obj from the top of stack to base-loc. Nothing is returned.

POP-M-FROM-UNDER-N num-pops, num-to-keep
[13] AUX-OP
While keeping the top num-to-keep values on the stack,

removes the num-pops values from underneath them.

POP-PDL-* [500-517] AUX-OP
Removes the top * elements from the PDL (stack cache).
* may be from 1 to 63.

POPJ return-macropc [14] AUX-OP
Transfer macrocontrol to return-pc. return-pc
is relative to the beginning of the FEF.
Therefore, the next macro-instruction executed will
be at return-pc.

PREDICATE [76] MISC-OP
Returns NIL if and only if the indicators are
null, otherwise return T.

PROPERTY-CELL-LOCATION symbol [634] MISC-OP Lisp-Functio
Returns a locative to symbol's property-list cell.
If symbol is not a symbol then the ARGTYP error
will be signalled.

PUSH base-obj [50] MAIN-OP
Pushes the value at base-obj to the
top of stack.

PUSH-AR-1 index, base-array [67] MAIN-OP
Pushes the value at the specified
index into the array at base-array.

PUSH-AREFI array, immed-index [47] AREFI
Returns the value of array specified by immed-index.
This is used for single-dimension array references
where immed-index is a small constant.
immed-index is an immediate value in the
instruction and the array is popped off the
stack.

PUSH-CADDR base-list [55] MAIN-OP
Pushes the CADDR of the list at base-list.

PUSH-CADR base-list [53] MAIN-OP
Pushes the CADR of the list at base-list.

PUSH-CAR base-list [51] MAIN-OP
Pushes the CAR of the list at base-list.

PUSH-CDDR base-list [54] MAIN-OP
Pushes the CDDR of the list at base-list.

PUSH-CDR base-list [52] MAIN-OP
Pushes the CDR of the list at base-list.

PUSH-CDR-STORE-CAR-IF-CONS x,base-dest [147] MAIN-OP

Returns the CDR of x (popped from stack)
and stores its CAR at base-dest if x is a cons,
otherwise NIL is left in the indicators.
This is used mainly to implement DOLIST.

PUSH-CONS car, base-cdr [56] MAIN-OP
Pushes the cons of car and the cdr at
base-cdr. Car is popped from the stack.

PUSH-GET sym, base-ind [57] MAIN-OP
Pushes sym's base-ind property, otherwise NIL
is returned. Returns NIL if sym does not have a
property list or is of a type that does not have
properties. Sym may be a symbol, list, locative,
or an instance. This instruction corresponds to
the GET function with two arguments.

PUSH-LOC base-loc [150] MAIN-OP
Pushes a locative that points to base-loc.

PUSH-LONG-FEF base-x [70] MAIN-OP
Pushes what is at base-x plus the current
FEF base address. base-x is a 9-bit offset.
The current FEF base address is defined to be the
function object of the currently executing stack
frame. The read of the object at base-x plus FEF
will be transported. This reference, therefore,
can cause a TRANS-TRAP error to be signalled if
the word read contains an invalid object or is
unbound.

PUSH-NEG-NUMBER [46] MAIN-OP
PUSH-NUMBER [47] MAIN-OP
Pushes the specified thing onto the stack. Knowing
what the thing is means greater efficiency because
these things never have to be TRANSPORTed.
These take an immediate operand which is used as
a 9-bit integer.

RATIONALP x [256] MISC-OP Lisp-Function
Returns T if x is a ratio or an integer,
otherwise returns NIL for all other data types.
An error is never signalled.

RATIOP x [257] MISC-OP Lisp-Function
Returns T if x is a ration, otherwise return NIL.
An error is never signalled.

RETURN base-val [17] MAIN-OP
Causes the current call frame to fold and a
return to the previous call frame. The value
at base-val will be left on top of the stack,
which is exactly where the current (function

executing RETURN) call frame started.

RETURN-* val0, val1, ..., val* [600-637] AUX-OP
Causes the current call frame to fold and a return to the previous call frame. *-values will be left on top of the stack, which is exactly where the current (function executing RETURN-*) call frame started. Therefore, this instruction moves the top * values to the new top of stack. Actually no more values are returned than the caller is expecting so the number returned may be less. * may be from 0 to 63 (decimal).

RETURN-LIST list [121] AUX-OP
Causes the current call frame to fold and a return to the previous call frame. Returns the elements of list as multiple values. This instruction will return the exact number of elements that the caller expects, i.e., if the number of elements in list is less than the number the caller expected to be returned, then NILs are used to pad. Conversely, the caller never receives more values than it is expecting.

RETURN-N val1 val2 ... valnumvals numvals
[120] AUX-OP
Exactly like RETURN-* except the number of values to return is specified with numvals. Again, the caller will never receive more values than it is expecting so the number returned may be less than numvals.

RETURN-NIL [122] AUX-OP
Causes the current call frame to fold and a return to the previous call frame. The value NIL will be left on top of the stack, which is exactly where the current (function executing RETURN-NIL) call frame started.

RETURN-NOT-INDS [137] AUX-OP
Causes the current call frame to fold and a return to the previous call frame. The value left on top of the stack, which is exactly where the current (function executing RETURN-NOT-INDS) call frame started will be T if the NIL indicator is not set. Otherwise NIL is returned.

RETURN-PRED [136] AUX-OP
Exactly like RETURN-NOT-INDS except returns T if the NIL indicator is set, otherwise returns NIL.

RETURN-T [123] AUX-OP
Causes the current call frame to fold and a return to the previous call frame. The value T will be left on top of the stack, which is exactly where the current (function executing RETURN-T) call frame started.

ROT n, nbits [532] MISC-OP Lisp-Function
Returns n rotated by nbits. The direction of rotation is controlled by the sign of nbits. If nbits is positive then a left shift is performed, otherwise a right shift is performed. n is treated as a 25-bit number and n and nbits must both be FIXNUMs.

ROUND-1 dividend, divisor [553] MISC-OP
Returns dividend divided by divisor, rounded to the nearest integer. The remainder is not returned.

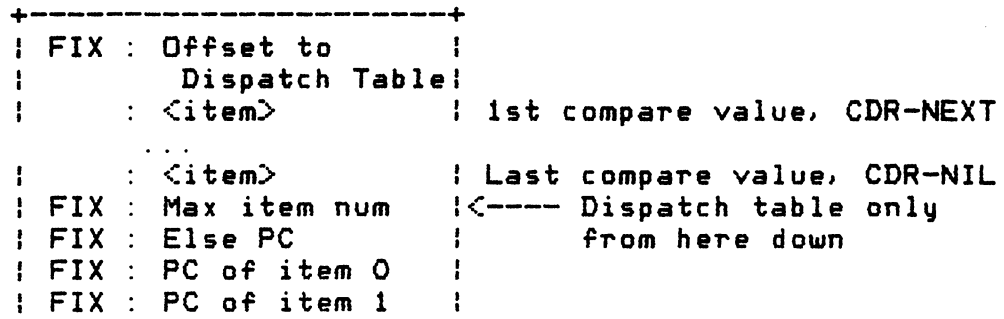
ROUND-2 dividend, divisor [557] MISC-OP
Returns dividend divided by divisor, rounded to the nearest integer and the remainder.

RPLACA cons, newcar [300] MISC-OP Lisp-Function
Returns cons with its CAR replaced with newcar. cons must be a list or a locative or the ARGTYP error is signalled.

RPLACD cons, newcdr [301] MISC-OP Lisp-Function
Returns cons with its CDR replaced with newcdr. cons must be a list or the ARGTYP error is signalled.

SCALE-FLOAT flonum, integer [545] MISC-OP Lisp-Function
Returns flonum with integer added to its exponent.

SELECT x, selectq-table [71] MAIN-OP
Used to implement multi-way branches based on value of x. Branches are defined by selectq-table which has the format shown below. It is a cdr-coded list of objects or EVCPs to objects (cdr coding must be correct).



```
        | FIX : PC of item 2    |
                 ...
        | FIX : PC of item n    |
        +-----------------------+
```

SET symbol, value [310] MISC-OP Lisp-Functio

Sets the value cell of symbol to value. Signals the ARGTYP error if symbol is not a symbol or is NIL (since it is illegal to set NIL).

SET-%INSTANCE-REF instance, index, value [354] MISC-OP

Sets the slot at index into the instance data structure to value.

SET-AR-1 array, subscript, value [331] MISC-OP Lisp-Functio

Sets the element of the one-dimensional array array specified by subscript to value. Array must be a one-dimensional array and subscript must be a FIXNUM. If index is less than zero or greater than the largest index permissible then a SUBSCRIPT-OOB error is signalled. If array is not one-dimensional then ARRAY-NUMBER-DIMENSIONS is signalled.

SET-AR-1-FORCE array, subscript, value [335] MISC-OP Lisp-Functio

Sets the element of the array array specified by subscript to value. Array is treated (forced) as a one-dimensional array; i.e., it is indexed with a single subscript regardless of its actual rank.

SET-AR-2 array, subscript1, subscript2, value [332] MISC-OP Lisp-Functio

Exactly like SET-AR-1 except array must be two-dimensional.

SET-AR-3 array subscript1 subscript2 subscript3 value [334] MISC-OP Lisp-Functio

Exactly like SET-AR-1 except array must be three-dimensional.

SET-AREF array, &REST subscripts-and-values UCODE ENTRY

SET-ARRAY-LEADER array index value [330] MISC-OP Lisp-Functio

Sets the array leader element of array specified by index to value. If array is not an array an ARGTYP error is signalled. If array does not have a leader the ARRAY-HAS-NO-LEADER error is signalled. If index is not a FIXNUM the ARGTYP error is signalled. Finally, if index is greater than or equal to the length of the leader the SUBSCRIPT-OOB error is signalled.

SET-NIL base-loc [155] MAIN-OP
Sets the value at base-loc to NIL.

SET-T base-loc [156] MAIN-OP
Sets the value at base-loc to T.

SET-ZERO base-loc [157] MAIN-OP
Sets the value at base-loc to a FIXNUM zero.

SETCAR cons, newcar [302] MISC-OP Lisp-Functio
Replaces the CAR of cons with newcar and returns newcar. Compare with RPLACA which returns a different value.

SETCDR cons, newcdr [303] MISC-OP Lisp-Functio
Replaces the CDR of cons with newcdr and returns newcdr. Compare with RPLACD which returns a different value.

SETE-1+ base-loc [144] MAIN-OP
Increments the contents at base-loc and stores the result back at base-loc. This is called a read-modify-write instruction. The contents at base-loc must be of the numeric type or an ARGTYP error is signalled.

SETE-1- base-loc [145] MAIN-OP
Exactly like SETE-1- except the contents at base-loc is decremented.

SETE-CDDR base-loc [143] MAIN-OP
Replaces base-loc with the CDDR of base-loc. This is called a read-modify-write instruction. (setq frob (cddr frob)) translates to this instruction. The contents at base-loc and its CDR must be of the list type or the ARGTYP error is signalled.

SETE-CDR base-loc [142] MAIN-OP
Replaces base-loc with the CDR of base-loc. This is called a read-modify-write instruction. (setq frob (cdr frob)) translates to this instruction. The contents at base-loc must be of the list type or the ARGTYP error is signalled.

SETELT sequence, index, value [307] MISC-OP Lisp-Functio
Sets the element of sequence at index to value. This corresponds to SETF on ELT.

SHRINK-PDL-SAVE-TOP n-slots value-to-move
[41] MISC-OP
Pops n-slots (after popping the 2 arguments above) from the PDL and moves value-to-move to the new top of the stack.

SIMPLE-ARRAY-P object [246] MISC-OP Lisp-Functio
Returns T if object is a simple array, otherwise

returns NIL. A simple array is an array that does not have a fill-pointer and is not displaced or indirect. This is a Common Lisp instruction.

SIMPLE-BIT-VECTOR-P object [251] MISC-OP Lisp-Function
Returns T if object is a simple bit vector, otherwise returns NIL. A simple bit vector is a one-dimensional array of array type ART-1B with no fill-pointer that is also not displaced or indirect.
This is a Common Lisp instruction.

SIMPLE-STRING-P object [247] MISC-OP Lisp-Function
Returns T if object is a simple string, otherwise returns NIL. A simple string is a one-dimensional array of array type ART-STRING or ART-FAT-STRING with no fill-pointer that is also not displaced or indirect. This is a Common Lisp instruction.

SIMPLE-VECTOR-P object [245] MISC-OP Lisp-Function
Returns T if object is a simple vector, otherwise returns NIL. A simple vector is a one-dimensional numeric array with no fill-pointer that is also not displaced or indirect. This is a Common Lisp instruction.

SMALL-FLOAT number [511] MISC-OP Lisp-Function
Converts number to a short float.

SMALL-FLOATP object [253] MISC-OP Lisp-Function
Returns T if object is of type SHORT-FLOAT (small-float), otherwise returns NIL. (Small-float is Zetalisp terminology, while short-float is the Common Lisp type name.)

SPECIAL-PDL-INDEX [405] MISC-OP Lisp-Function
Returns a locative pointing to the last slot of the current special PDL that was bound.

STACK-GROUP-RESUME sg, x [47] MISC-OP Lisp-Function
Resumes the stack group sg with the argument x.
See the stack groups section.

STACK-GROUP-RETURN x [46] MISC-OP Lisp-Function
Resume the stack group which invoked the current stack group with the argument x. The resumed stack group's resumer does not change.

STORE-ARRAY-LEADER value array index [320] MISC-OP Lisp-Function
Sets the array leader element of array specified by index to value. If array is not an array, an ARGTYP error is signalled. If array does not have a leader the ARRAY-HAS-NO-LEADER error is signalled. If index is not a FIXNUM, the ARGTYP error is signalled. Finally, if index is greater than or

equal to the length of the leader, the SUBSCRIPT-OOB error is signalled.

STORE-IN-HIGHER-CONTEXT value context-desc [30] AUX-OP
Used for storing into lexical variables in a higher lexical context.

STRINGP base-x [33] MAIN-OP Lisp-Function
x [601] MISC-OP Lisp-Function
The STRINGP MAIN-OP sets the indicators if base-x is a one-dimensional array of array-type ART-STRING or ART-FAT-STRING. The STRINGP MISC-OP returns T if x is a one-dimensional array of array-type ART-STRING or ART-FAT-STRING, otherwise return NIL.

SYMBOL-FUNCTION symbol [627] MISC-OP Lisp-Function
This is exactly like FSYMEVAL (same MISC-OP number, also). The difference is that the functionality is known in the Lisp world under this name.

SYMBOL-NAME symbol [631] MISC-OP Lisp-Function
Returns the print name of symbol (an array pointer). Symbol must be a symbol or the ARGTYP error is signalled. This was called GET-PNAME in Zetalisp.

SYMBOL-PACKAGE symbol [635] MISC-OP Lisp-Function
Returns the package object of symbol. Symbol must be a symbol or the ARGTYP error is signalled.

SYMBOL-VALUE symbol [636] MISC-OP Lisp-Function
Returns the current value of symbol. Symbol must be a symbol or the ARGTYP error is signalled. If symbol is unbound then the TRANS-TRAP error is signalled. This was called SYMEVAL in Zetalisp.

SYMBOLP x [577] MISC-OP Lisp-Function
Returns T if x is a symbol, otherwise NIL is returned.

TEST base-obj [10] MAIN-OP
Sets the indicators based on the contents of base-obj.

TEST-AREFI [7] AREF1
Sets the indicators based on the contents of an immediate array element reference.

TEST-CAAR base-list [15] MAIN-OP
Sets the indicators based on the contents of the CAAR of base-list.

TEST-CADR base-list [13] MAIN-OP
Sets the indicators based on the contents of the CADR of base-list.

TEST-CAR base-list [11] MAIN-OP
Sets the indicators based on the contents of the CAR of base-list.

TEST-CDDR base-list [14] MAIN-OP
Sets the indicators based on the contents of the CDDR of base-list.

TEST-CDR base-list [12] MAIN-OP
Sets the indicators based on the contents of the CDR of base-list.

TEST-MEMQ x, base-list [16] MAIN-OP
Sets the indicators based on either the sublist of base-list beginning with the first occurrence of x (using EQ), if found, or NIL.

TIME-IN-60THS [472] MISC-OP Lisp-Functio
Used internally to implement TIME when called with no arguments.

TRUNCATE-1 dividend, divisor [552] MISC-OP
Returns dividend divided by divisor, rounded down to zero (truncated). The remainder is not returned.

TRUNCATE-2 dividend, divisor [556] MISC-OP
Returns dividend divided by divisor, rounded down to zero (truncated) and the remainder.

TYPEP-STRUCTURE-OR-FLAVOR object, type [252] MISC-OP Lisp-Functio
Used for TYPEP when the Compiler knows that the type being tested for is a flavor or named structure.

UNBIND-* [400-417] AUX-OP
Undo * bindings on the special PDL (binding stack). * may be from 1 to 63.

UNBIND-TO-INDEX special-pdl-index [17] AUX-OP
Undo bindings on the special PDL (binding stack) until the special PDL pointer is less than or equal to special-pdl-index.

UNBIND-TO-INDEX-MOVE [406] MISC-OP
special-pdl-index, value-to-move
Exactly like UNBIND-TO-INDEX, except value-to-move is returned.

VALUE-CELL-LOCATION symbol [632] MISC-OP Lisp-Functio
Returns a locative to the internal value cell of

symbol. Symbol must be a symbol or the ARGTYP error is signalled. Note that this instruction will return a pointer to the exact contents of the value cell and will not follow forwarding pointers.

VECTOR-PUSH new-element, vector [341] MISC-OP Lisp-Functio
Returns the new fill-pointer for vector after pushing new-element, otherwise returns NIL if vector is full. Vector must have a leader or the ARRAY-HAS-NO-LEADER error is signalled. The fill-pointer of vector (leader element 0) must be a FIXNUM or the FILL-POINTER-NOT-FIXNUM error is signalled. This instruction does not check that vector is a true vector.
This is a Common Lisp instruction.

VECTORP object [244] MISC-OP Lisp-Functio
Returns T if object is a vector, otherwise returns NIL. A vector is a one-dimensional array.
This is a Common Lisp instruction.

ZEROP number [560] MISC-OP Lisp-Functio
Returns T if number is equal to the zero of its type, otherwise returns NIL. Number must be of type Number (numeric) or the ARGTYP error is signalled.

%ADD-INTERRUPT device-desc, level [211] MISC-OP
Installs an interrupt for the device described by the array device-desc at the level level. The device encoded within device-desc must have a microcode interrupt handler for it or an error is signalled. This instruction does not perform any device initializations.

%ALLOCATE-AND-INITIALIZE data-type header-type header second-word area size
[415] MISC-OP Lisp-Functio
This is the subprimitive for creating most structured-type objects. Area is the area in which it is to be created, as a FIXNUM or a symbol. Size is the number of words to be allocated. The value returned points to the first word allocated and is of type data-type. The words allocated are initialized with interrupts disallowed so that storage conventions are preserved at all times. The first word, the header, is initialized to have header-type in its data-type field and header in its pointer field. The second word is initialized to second-word. The remaining words are initialized to NIL. The cdr-codes of all words except the last are set to cdr-next; the cdr-code

of the last word is set to cdr-nil. Note that programs should not rely on the cdr-code field of non-cons cells being in a known state.

%ALLOCATE-AND-INITIALIZE-ARRAY header index-length leader-length area nqs
[416] MISC-OP Lisp-Function

This is the subprimitive for creating arrays, called only by make-array. It is different from %allocate-and-initialize because arrays have a more complicated header structure.

%ALLOCATE-AND-INITIALIZE-INSTANCE header, area, nqs
[420] MISC-OP Lisp-Function

Allocates storage for an instance, sets header type to DTP-Instance-Header and sets data type to DTP-Instance. Fills allocated space with NIL and places header in word 0.

%AREA-NUMBER x [500] MISC-OP Lisp-Function

Returns the area number of the area the pointer x points into, or NIL.

%ASSURE-PDL-ROOM room [12] AUX-OP Lisp-Function

This instruction will trap if there are not room more words available in this function frame.

%BLT from-address, to-address, count, increment
[346] MISC-OP Lisp-Function

Copy a block of virtual memory, a word at a time, with no decoding, for untyped data. Use %BLT-TYPED for words which contain Lisp data types. The first word is copied from from-address to to-address. Increment is added to each address and then another word is copied till count is exhausted.

%BLT-FROM-PHYSICAL source-address destination-address, number-of-words increment
[224] MISC-OP Lisp-Function

Copy a block of physical memory from source-address to unboxed virtual memory starting at destination-address. Not decoded; use on untyped data only.

%BLT-TO-PHYSICAL source-address destination-address number-of-words increment
[223] MISC-OP Lisp-Function

Copy a block of physical memory from unboxed virtual memory starting at source-address to physical memory beginning at destination-address. Not decoded; use on untyped data only.

%BLT-TYPED [347] MISC-OP Lisp-Function

Copy a block of virtual memory, a word at a time.

The first word is copied from from-address to to-address. Increment is added to each address and then another word is copied till count is exhausted. Each word copied is transported and each word written is checked through the the Write Barrier. Returns NIL.

%CHANGE-PAGE-STATUS virt-addr swap-status access-and-meta [360] MISC-OP Lisp-Function
Changes the page status bits of the page containing virt-addr, if it is paged in, to swap-status and access-and-meta. If either swap-status or access-and-meta are NIL then that parameter is not set. Returns T if the page was found in the page hash table (it was swapped in) or NIL if it was not found (it was not swapped in). This does no error checking.

%CLOSE-CATCH [134] AUX-OP
Close a catch.

%CLOSE-CATCH-UNWIND-PROTECT [135] AUX-OP
Close catch but leave info for %unwind-protect-continue.

%COMPUTE-PAGE-HASH addr [475] MISC-OP Lisp-Function
Computes the page hash table index that corresponds to addr and returns it as a FIXNUM.

%CRASH code, object, paws-up-p [3] AUX-OP Lisp-Function
Causes machine to crash (like ILLOP) indicating crash reason as software with code remembered as the crash code. Object is also remembered for crash analysis. If paws-up-p is not NIL then will display paws-up as ILLOP does. If paws-up-p is NIL then it is presumed that the called has indicated the lossage to the user. %CRASH is not restartable, but you may be able to warm-boot out of it.

%CREATE-PHYSICAL-PAGE pfn [140] AUX-OP Lisp-Function
Adds the physical page specified by pfn to the pool of available page frames. See section on Paging and Disk Management for more details.

%DATA-TYPE x [450] MISC-OP Lisp-Function
Returns the data type of x as a FIXNUM. Its value will be less than 32.

%DELETE-PHYSICAL-PAGE pfn [362] MISC-OP Lisp-Function
Deletes the physical page specified by pfn from the page frame pool. Returns T if successful, otherwise NIL. See section on Paging and Disk Management for more details.

%DISK-RESTORE [156] AUX-OP Lisp-Functio
partition-high-16-bits low-16-bits physical-unit
Restores the load partition whose name is formed by concatenating the first two arguments to form a 32-bit number (4 characters). If the number is zero then the default band is restored from the default unit, else the named partition is restored from physical-unit.

%DIV dividend, divisor [544] MISC-OP Lisp-Functio
Returns the rational number generated by dividing dividend by divisor.

%DRAW-CHAR font-array char-code x-bitpos y-bitpos alu-function sheet
[0] MODULE(TV)
Draws the character char-code of font font-array on sheet using alu-function. alu-function is typically TV:ALU-IOR, TV:ALU-ANDCA, or TV:ALU-XOR. x-bitpos and y-bitpos are the position in sheet for the upper left corner of the character to be drawn.

%DRAW-FILLED-RASTER-LINE x1 x2 y left-edge top-edge right-edge bottom-edge alu draw-last-point fill-color destination
[5] MODULE(TV)

%DRAW-FILLED-TRIANGLE x1 y1 x2 y2 x3 y3 left-edge top-edge right-edge bottom-edge alu draw-third-edge draw-second-edge draw-first-edge fill-color destination
[4] MODULE(TV)

%DRAW-LINE x0 y0 x y alu draw-end-point sheet
[2] MODULE(TV)
Draws a straight line from (x0,y0) to (x,y) on sheet using alu as the ALU function. alu is typically TV:ALU-IOR, TV:ALU-ANDCA, or TV:ALU-XOR.

%DRAW-RECTANGLE width height x-bitpos y-bitpos alu-function sheet
[1] MODULE(TV)
Draws a solid rectangle on sheet using alu-function. alu is typically TV:ALU-IOR, TV:ALU-ANDCA, or TV:ALU-XOR. Height and width are the size of the rectangle, and x-bitpos and y-bitpos are the location of the upper left corner.

%EXTERNAL-VALUE-CELL symbol [637] MISC-OP Lisp-Functio
Returns a locative to whatever the value cell of symbol points to. If symbol is closure bound, this will be a locative to the external value cell. Does not check that the internal value cell contains an external value cell pointer.

%FIND-STRUCTURE-HEADER ptr [502] MISC-OP Lisp-Functio

Returns the object containing the cell addressed by the locative, ptr. Finds the overall structure containing the cell addressed by ptr. Does not follow structure forwarding.

%FINDCORE [476] MISC-OP Lisp-Functio
Returns the page frame number of an available physical. page. Makes one available if necessary..

%FIXNUM-MICROSECOND-TIME [471] MISC-OP Lisp-Functio
Returns the 32-bit microsecond time truncated to 25-bits and typed as a FIXNUM.

%FUNCTION-INSIDE-SELF [714] MISC-OP Lisp-Functio
Returns the functional part of SELF. If SELF is an instance, return the contents of the cell referenced by the %INSTANCE-DESCRIPTOR-FUNCTION slot of the instance descriptor. This is usually a funcallable hash array. If SELF is a closure, return the function from the closure, otherwise return NIL.

%GC-CONS-WORK nqs [153] AUX-OP Lisp-Functio
Informs the GC microcode that nqs Q's have been allocated. There is no need to do this if storage is allocated by the microcoded storage allocation routines.

%GC-FLIP region [151] AUX-OP Lisp-Functio
Flips region converting new space to old space. Ensures that nothing else in the machine points to old space. If region is T all new space is converted to old space.

%GC-FREE-REGION region [150] AUX-OP Lisp-Functio
Makes region region free. Used on old space region after scavenging is complete.

%GC-SCAV-RESET region [57] MISC-OP
Returns T if the scavenger was looking at this region, otherwise NIL is returned. This also makes the scavenger not look at region and removes region from the cons cache.

%GC-SCAVENGE work-units [152] AUX-OP Lisp-Functio
Scavenge for work-units of work or until a page fault. Returns NIL if completed work-units of work or ran out of work to do. Returns non-NIL if took a page fault before scavenging was complete. A "work-unit" is the scavenging of one Q.

%GET-SELF-MAPPING-TABLE method-flavor-name
[711] MISC-OP Lisp-Functio
Returns NIL if SELF is not an instance.
Returns the value of SELF-MAPPING-TABLE if the mapping table is already this value.
Otherwise, returns the table located by searching the

mapping table alist of the instance descriptor for SELF for method-flavor-name and getting its CDDR. method-flavor-name is a symbol for the flavor of the method for which to get a self mapping table.

%INSTANCE-LOC instance, index [713] MISC-OP Lisp-Function
Returns a locative to the slot index in instance. The lowest valid index is 1 (1-origin).

%INSTANCE-REF instance, index [712] MISC-OP Lisp-Function
Returns the contents of the slot index in instance. The lowest valid index is 1 (1-origin).

%IO rqb, device-desc [210] MISC-OP Lisp-Function
Initiate IO request describe rqb (request-block) on the device described by the array device-desc. The device handler will interpret the contents of the rqb. Rqb is often an array but may be a FIXNUM. device-desc is an array that serves as the IO-DEVICE-DESCRIPTOR (see section on devices).

%LOGDPB value, ppss, word [525] MISC-OP Lisp-Function
A FIXNUMs-only form of DPB. The low order ss bits of value replace the field of word. Always returns a FIXNUM. Does not complain about loading/clobbering the sign bit. No error checking is performed on the arguments.

%LOGLDB ppss, word [521] MISC-OP Lisp-Function
A FIXNUMs-only form of LDB. Returns a FIXNUM obtained from the 32-bit uninterpreted word. This instruction only return a field up to 25 bits. The result may be negative if the field size is 25. Signals the ARGTYP error if ppss is not a FIXNUM or if it specifies a field greater than 25 bits wide.

%MAKE-EXPLICIT-STACK-LIST length [401] MISC-OP Lisp-Function
Returns a list pointer to the first element of the list immediately before the argument length. The length values prior to length are made into a list; i.e., the CDR-code of the last one is changed to CDR-NIL.

%MAKE-EXPLICIT-STACK-LIST* length [402] MISC-OP Lisp-Function
Exactly the same as %MAKE-EXPLICIT-STACK-LIST except it makes the last element be the cdr of the list.

%MAKE-LIST initial-value, area, length [414] MISC-OP Lisp-Function
Constructs a CDR-coded list of initial-value, length elements long in area.

%MAKE-POINTER dtp, address [446] MISC-OP Lisp-Function

Returns a Q whose data type is dtp and whose pointer field is address; exercise extreme caution.

%MAKE-POINTER-OFFSET new-dtp, pointer, offset
[447] MISC-OP Lisp-Function
Returns a Q whose data type is dtp and whose pointer field is pointer added to offset; exercise extreme caution.

%MAKE-REGION bits, size [505] MISC-OP
Creates a region whose size is at least size and whose region bits are set to bits.

%MAKE-STACK-LIST n [400] MISC-OP Lisp-Function
Returns a pointer of data type DTP-STACK-LIST to the list constructed by pushing n NILS on the stack with CDR-NEXT on them all except the last one, which receives a CDR-NIL. This does not check for PDL room and therefore the execution of %ASSURE-PDL-ROOM before this is recommended.

%MICROSECOND-TIME [470] MISC-OP Lisp-Function
Returns the 32-bit microsecond time as an integer, either a FIXNUM or a BIGNUM.

%MULTIBUS-READ-* multibus-byte-adr [431-433] MISC-OP Lisp-Function
Signals the UNIMPLEMENTED-HARDWARE error.
* is 8, 16, or 32.

%MULTIBUS-WRITE-* multibus-byte-adr word
[214-216] MISC-OP Lisp-Function
Signals the UNIMPLEMENTED-HARDWARE error.
* is 8, 16, or 32.

%NUBUS-READ NuBus-slot slot-byte-adr [434] MISC-OP Lisp-Function
Returns a signed integer, either a FIXNUM or a BIGNUM, read from the word (32-bits) at the physical address formed by taking the low order 24-bits from slot-byte-adr and concatenating the low 8-bits from NuBus-slot. For NuBus slot space accesses, the 4-bits at bit 4 of NuBus-slot should be ones. If a NuBus error occurs, then the USER-NUBUS-ERROR error is signalled. If a GACBL (Go Away Come Back Later) is encountered, it is retried. After a very large number of retries, an error is assumed and the USER-NUBUS-ERROR is signalled.

%NUBUS-READ-* hi-address low-address
[435-436] MISC-OP Lisp-Function
Returns a FIXNUM read from the byte (when * is 8b) or halfword (when * is 16B) at the physical address formed by taking the low order 24-bits from low-address and concatenating the low 8-bits from hi-address.

For NuBus slot space accesses, the 4-bits at bit 4 of low-address should be ones. If a NuBus error occurs then the USER-NUBUS-ERROR error is signalled. If a GACBL (Go Away Come Back Later) is encountered, it is retried. After a very large number of retries, an error is assumed and the USER-NUBUS-ERROR is signalled.

%NUBUS-READ-8B-CAREFUL hi-address low-address
[440] MISC-OP Lisp-Functio

Returns a FIXNUM read from the byte at the physical address formed by taking the low order 24-bits from low-address and concatenating the low 8-bits from hi-address. For NuBus slot space accesses, the 4-bits at bit 4 of low-address should be ones. If a NuBus error occurs then NIL is returned. If a bus timeout occurs then T is returned. If a GACBL (Go Away Come Back Later) is encountered, it is retried. After a very large number of retries, an error is assumed and NIL is returned.

%NUBUS-WRITE NuBus-slot slot-byte-adr word
[217] MISC-OP Lisp-Functio

Writes word (32-bits) at the physical address formed by taking the low order 24-bits from slot-byte-adr and concatenating the low 8-bits from NuBus-slot. For NuBus slot space accesses, the 4-bits at bit 4 of NuBus-slot should be ones. Word should be a signed FIXNUM or a BIGNUM. If a NuBus error occurs then the USER-NUBUS-ERROR error is signalled. If a GACBL (Go Away Come Back Later) is encountered, it is retried. After a very large number of retries, an error is assumed and the USER-NUBUS-ERROR is signalled.

%NUBUS-WRITE-* hi-address low-address data
[220-221] MISC-OP Lisp-Functio

Writes word, a FIXNUM byte (if * is 8b) or a FIXNUM halfword (when * is 16b), at the physical address formed by taking the low order 24-bits from slot-byte-adr and concatenating the low 8-bits from NuBus-slot. For NuBus slot space accesses, the 4-bits at bit 4 of NuBus-slot should be ones. If a NuBus error occurs then the USER-NUBUS-ERROR error is signalled. If a GACBL (Go Away Come Back Later) is encountered, it is retried. After a very large number of retries, an error is assumed and the USER-NUBUS-ERROR is signalled.

%OPEN-CATCH catch-tag, restart-pc [124] AUX-OP
%OPEN-CATCH-MULTIPLE-VALUE catch-tag restart-pc number-of-values
[125] AUX-OP
%OPEN-CATCH-MV-LIST catch-tag, restart-pc

```
                                                [127]   AUX-OP
%OPEN-CATCH-TAIL-RECURSIVE  catch-tag restart-pc
                                                [126]   AUX-OP
        Opens a catch block data structure on the stack.
        Catch-tag and restart-pc are some of the elements
        of that data structure.  See the discussion on
        catching and throwing.

%OPEN-MOUSE-CURSOR                               [1]  MODULE(MOUSE)

%P-CDR-CODE  pointer                            [451]  MISC-OP  Lisp-Function
%P-CONTENTS-AS-LOCATIVE  pointer                [461]  MISC-OP  Lisp-Function
%P-CONTENTS-AS-LOCATIVE-OFFSET pointer offset
                                                [462]  MISC-OP  Lisp-Function
%P-CONTENTS-OFFSET  pointer offset              [456]  MISC-OP  Lisp-Function
%P-DATA-TYPE  pointer                           [452]  MISC-OP  Lisp-Function
%P-DEPOSIT-FIELD value ppss pointer              [51]  MISC-OP  Lisp-Function
%P-DEPOSIT-FIELD-OFFSET  value ppss pointer offset
                                                 [53]  MISC-OP  Lisp-Function
%P-DPB  value ppss pointer                       [50]  MISC-OP  Lisp-Function
%P-DPB-OFFSET  value ppss pointer offset        [52]  MISC-OP  Lisp-Function
%P-LDB  ppss pointer                            [454]  MISC-OP  Lisp-Function
%P-LDB-OFFSET  ppss pointer offset              [457]  MISC-OP  Lisp-Function
%P-MASK-FIELD  ppss pointer                     [455]  MISC-OP  Lisp-Function
%P-MASK-FIELD-OFFSET  ppss pointer offset
                                                [456]  MISC-OP  Lisp-Function
%P-POINTER  pointer                             [453]  MISC-OP  Lisp-Function
%P-STORE-CDR-CODE  pointer cdr-code              [64]  MISC-OP  Lisp-Function
%P-STORE-CONTENTS  pointer value                 [62]  MISC-OP  Lisp-Function
%P-STORE-CONTENTS-OFFSET  value pointer offset
                                                 [67]  MISC-OP  Lisp-Function
%P-STORE-DATA-TYPE  pointer data-type            [65]  MISC-OP  Lisp-Function
%P-STORE-POINTER  pointer pointer-to-store
                                                 [66]  MISC-OP  Lisp-Function
%P-STORE-TAG-AND-POINTER  pointer misc-fields pointer-field
                                                 [63]  MISC-OP  Lisp-Function
        These pointer-manipulation miscops and other
        Lisp-coded ones are fully described in the
        section on Storage Subprimitives.

%PAGE-IN  pfn vpn                               [363]  MISC-OP  Lisp-Function
%PAGE-STATUS  ptr                               [474]  MISC-OP  Lisp-Function
%PAGE-TRACE  table                              [371]  MISC-OP  Lisp-Function
%PHYSICAL-ADDRESS ptr                           [430]  MISC-OP  Lisp-Function
        See the section on Paging and Disk Management
        for description.

%POINTER x                                      [445]  MISC-OP  Lisp-Function
        Returns the pointer field of x as a FIXNUM.

%POINTER-DIFFERENCE ptr1 ptr2                   [463]  MISC-OP  Lisp-Function
        Returns the number of words between pointers
        ptr2 and ptr1.  These pointers can be anything,
```

but are usually FIXNUMs or locatives. This number can change due to GC.

%RATIO-CONS numerator denominator [1037] MISC-OP Lisp-Function
Returns a rational number constructed from numerator and denominator.

%RECORD-EVENT data-4 data-3 data-2 data-1 stack-level event must-be-4
[373] MISC-OP Lisp-Function

%REGION-NUMBER x [660] MISC-OP Lisp-Function
Returns the region number object x points into, or NIL.

%SET-MOUSE-SCREEN window [0] MODULE-OP
Sets the current screen for the mouse.

%SET-SELF-MAPPING-TABLE mapping-table [22] AUX-OP

%SPREAD list [11] MISC-OP Lisp-Function
Takes a list and pushes its elements on the stack.

%STACK-FRAME-POINTER [404] MISC-OP Lisp-Function
Returns a locative pointing at the current stack frame. This happens to be the same as a pointer to local 0.

%STORE-CONDITIONAL pointer old new [464] MISC-OP Lisp-Function
Store new into pointer if the old contents of pointer match old. Returns T if the store was done, otherwise NIL. This is a basic interlocking primitive, which can be used to simulate any sort of atomic test-and-modify operation.

%STRING-EQUAL string1 index1 string2 index2 count
[233] MISC-OP Lisp-Function
T if count characters of string1 at index1 match those of string2 at index2. Similar to STRING-EQUAL, but args are slightly different and all required -- and it's faster. The comparison ignores case unless ALPHABETIC-CASE-AFFECTS-STRING-COMPARISON is non-NIL.

%STRING-SEARCH-CHAR char string start end
[701] MISC-OP Lisp-Function
The same as STRING-SEARCH-CHAR, but without coercion and error checking. Also, all the args are required. And it's faster.

%STRING-WIDTH table offset string start end stop-width
[702] MISC-OP Lisp-Function

%STRUCTURE-BOXED-SIZE ptr [503] MISC-OP Lisp-Function
Returns the number of normal Lisp pointers in the object indicated by ptr. This many words at the beginning of the object contain normal Lisp data. The remaining words contain just numbers (such as the instructions of a compiled function, or the data in a numeric array).

%STRUCTURE-TOTAL-SIZE ptr [504] MISC-OP Lisp-Function
Returns the number of words in the object indicated by ptr.

%SXHASH-STRING string character-mask [700] MISC-OP Lisp-Function
Returns a hash code for string computed on a character-by-character basis after applying character mask to each character.

%TEST&SET-68K slot offset [227] MISC-OP
Test for 1 (then set it) in high bit of byte specified by slot and offset. Return T if bit is not already set, else NIL.

%THROW tag value [130] AUX-OP
%THROW-N tag &rest values-and-count [131] AUX-OP
Throw one or more values to tag. See discussion of throwing in the Function Calling section.

%UNWIND-PROTECT-CLEANUP [15] AUX-OP
%UNWIND-PROTECT-CONTINUE [132] AUX-OP
Maybe continue throwing after unwind-protect undo-forms.

%USING-BINDING-INSTANCES binding-instances
[16] AUX-OP Lisp-Function
See section on subprimitives for description.

%WRITE-INTERNAL-PROCESSOR-MEMORIES code adr d-hi d-low
[370] MISC-OP Lisp-Function

*BOOLE fn arg1 arg2 [533] MISC-OP
- x y [61] MAIN-OP
LOGAND x y [63] MAIN-OP
LOGIOR x y [542] MISC-OP
LOGXOR x y [64] MAIN-OP
MAX num1 num2 [534] MISC-OP
MIN num1 num2 [535] MISC-OP
+ x y [60] MAIN-OP
QUOTIENT num1 num2 [541] MISC-OP
* x y [62] MAIN-OP
Limited-argument instructions and miscop forms of common functions. See descriptions of corresponding multi-argument Lisp functions.

*UNWIND-STACK tag value frame-count action [133] AUX-OP Lisp-Functio

Throw completely out of a computation, running all unwind protects found along the way. See section on function calling for details.

1+ n [65] MAIN-OP Lisp-Functio
1- n [66] MAIN-OP Lisp-Functio

Pushes on the stack the result of incrementing or decrementing the operand.

<-IMMED x immed-y [6] IMMEDIATE
=-IMMED x immed-y [4] IMMEDIATE
>-IMMED x immed-y [5] IMMEDIATE

Sets the indicators as a result of comparing a number popped from the stack to the 9-bit immediate operand treated as a signed integer.

< num1 num2 [570] MISC-OP
= num1 num2 [572] MISC-OP
> num1 num2 [571] MISC-OP

Limited-argument miscop forms of common functions. See descriptions of corresponding multi-argument Lisp functions.