Mesa 6.0 Primary Bytecodes

000 NOOP 001 ME 002 MRE 003 MXW
004 MXD 005 NOTIFY 006 BCAST 007 REQUEUE
008 LL0 009 LL1 010 LL2 011 LL3
012 LL4 013 LL5 014 LL6 015 LL7
016 LLB 017 LLDB 018 SL0 019 SL1
020 SL2 021 SL3 022 SL4 023 SL5
024 SL6 025 SL7 026 SLB 027 PL0
028 PL1 029 PL2 030 PL3 031 LG0
032 LG1 033 LG2 034 LG3 035 LG4
036 LG5 037 LG6 038 LG7 039 LGB
040 LGDB 041 SG0 042 SG1 043 SG2
044 SG3 045 SGB 046 LI0 047 LI1
048 LI2 049 LI3 050 LI4 051 LI5
052 LI6 053 LIN1 054 LINI 055 LIB
056 LIW 057 LINB 058 LADRB 059 GADRB
060 061 062 063
064 R0 065 R1 066 R2 067 R3
068 R4 069 RB 070 W0 071 W1
072 W2 073 WB 074 RF 075 WF
076 RDB 077 RD0 078 WDB 079 WD0
080 RSTR 081 WSTR 082 RXLP 083 WXLP
084 RILP 085 RILP 086 WILP 087 RIL0
088 WS0 089 WSB 090 WSF 091 WSDB
092 RFC 093 RFS 094 WFS 095
096 097 098 099
100 101 102 103
104 105 106 107
108 109 110 111
112 113 114 SLDB 115 SGDB
116 PUSH 117 POP 118 EXCH 119 LINKB
120 DUP 121 NILCK 122 123 BNDCK
124 125 126 127
128 J2 129 J3 130 J4 131 J5
132 J6 133 J7 134 J8 135 J9
136 JB 137 JW 138 JEQ2 139 JEQ3
140 JEQ4 141 JEQ5 142 JEQ6 143 JEQ7
144 JEQ8 145 JEQ9 146 JEQB 147 JNE2
148 JNE3 149 JNE4 150 JNE5 151 JNE6
152 JNE7 153 JNE8 154 JNE9 155 JNEB
156 JEB 157 JGB 158 JGB 159 JEB
160 JULB 161 JUVEB 162 JUGB 163 JULEB
164 J2EQB 165 J2NEB 166 167 JIW
168 ADD 169 SUB 170 MUL 171 DBL
172 DIV 173 LDIV 174 NEG 175 INC
176 ADD 177 OR 178 XOR 179 SHIFT
180 DADD 181 DSUB 182 DCOMP 183 DUCOMP
184 ADD01 185 186 187
188 189 190 191
192 EFC0 193 EFC1 194 EFC2 195 EFC3
196 EFC4 197 EFC5 198 EFC6 199 EFC7
200 EFC8 201 EFC9 202 EFC10 203 EFC11
204 EFC12 205 EFC13 206 EFC14 207 EFC15
Operations on the stack:

- **DIS**: Discard the top element of the stack (decrement the stack pointer)
- **REC**: Recover the previous top of stack (increment the stack pointer)
- **EXCH**: Exchange the top two elements of the stack
- **DEXCH**: Exchange the top two doubleword elements of the stack
- **DUP**: Duplicate the top element of the stack
- **DDUP**: Duplicate the top doubleword element of the stack
- **DBL**: Double to top of stack (multiply by 2)

 Unary operations: **NEG, INC, DEC**, etc.

 Logical operations: **IOR, AND, XOR**.

 Arithmetic: **ADD, SUB, MUL**.

 Doubleword arithmetic: **DADD, DSUB**.

 Divide and other infrequent operations are relegated to a multibyte escape opcode that extends the instruction set beyond 256 instructions.

 Simple Load and Store instructions:

- **LIN**: Load Immediate n
- **LIB a**: Load Immediate Byte
- **LIW aB**: Load Immediate Word
- **LLn**: Load Local n; load the word at offset n from LF
- **LLB a**: Load Local Byte; load the word at offset a from LF
SLn Store Local n
SLB a Store Local Byte
PLn Put Local n; equivalent to SLn REC, i.e. store and leave the value on the stack
LGn Load Global n; load the word at offset n from GF
LGB a Load Global Byte; load the word at offset a from GF
SGB a Store Global Byte
LLKB a Load Link; load a word at offset a in the link space

There are also versions of these instructions that load doubleword quantities. Note that there are no three-byte versions of these loads and stores and no one-byte Store Global instructions. These do not occur frequently enough to warrant inclusion in the instruction set.

Jumps:
All jump distances are measured in bytes relative to the beginning of the jump instruction; they are specified as signed eight or 16 bit numbers

Jn Short positive jumps
JB a jump -128 to + 127 bytes
JW aB long positive or negative jumps
JLB a compare (unsigned) top two elements of stack and jump if less; also JLEB, JEB, JGB, JGEB and unsigned versions
JEBB aB if top of stack is equal to a, jump distance in B; also JNBB
JZB a jump if top of stack is zero; also JNZB
JEP a if top of stack is equal to a.left, jump distance in a.right also JNEP
JIB aB at offset aB in the code segment find a table of eight bit distances to be indexed by the top of stack; also JIW with a table of sixteen bit distances.

Read and Write through pointers:
These instructions read and write data through pointers on the stack or stored in local variables

Rn Read through pointer on stack plus small offset
RB a Read through pointer on stack plus offset a
WB Write through pointer on stack plus offset a
RLIP a  Read Local Indirect; use pointer in local variable a.left
add offset a.right

WLIP a  Write local indirect

RnF a  Read Field using pointer on the stack plus n; a contains
starting bit and bit count as four bit quantities

RF aB  Read Field using pointer on the stack plus a; B contains
starting bit and bit count as four bit quantities

WF aB  Write Field

RKIB a  Read Link Indirect; use the word at offset a in the link
space as a pointer

Control Transfers:
These instructions handle procedure call return. Local calls (in the same
module) specify the entry point number of the destination procedure; external
calls (to another module) specify an index of a control link in the module’s
link space

LFCn  Local Function Call using entry point n
LFCB a Local Function Call using entry point a
EFCn  External Function Call using control link n
EFCB a External Function Call Byte using control link a
SFC  Stack Function Call; use control link from the stack
RET  Return. XFER using the return link in the local frame as the
destination; free the frame
BRK  Breakpoint; a distinguished one-byte instruction that causes a
trap

Miscellaneous:
These instructions are used to generate and manipulate pointer values

LAn  Local Address n; put the address of local variable n on the stack
LAB a Local Address Byte; put the address of local variable a on the
stack
LAW aB Local Address Word; put the address of local variable aB on the
stack
GAn  Global Address n; put the address of global variable n on the
stack
GAB a  Global Address Byte; put the address of global variable a on the
stack
GAW aB       Global Address Word; put the address of global variable aB on the stack

LP           Lengthen Pointer; convert the short pointer on the stack to a long pointer by adding MDS; includes a check for invalid pointers