IADD (ISUB, IAND, IOR similarly)

iadd1: MAR = SP = SP - 1; rd;

iadd2: H = TOS;

iadd3: MDR = MDR + H; wr; goto Main1;

DUP

dup1: MAR = SP = SP + 1;

dup2: MDR = TOS; wr; goto Main1;

POP

pop1: MAR = SP = SP – 1; rd;

pop2:

pop3: TOS = MDR; goto Main1;

SWAP

swap1: MAR = SP – 1; rd;

swap2: MAR = SP;

swap3: H = MDR; wr;

swap4: MDR = TOS;

swap5: MAR = SP – 1; wr;

swap6: TOS = H; goto Main1;

BIPUSH byte

bipush1: MAR = SP = SP + 1;

bipush2: PC = PC + 1; fetch;

bipush3: MDR = TOS = MBR; wr; goto Main1;

ILOAD varnum

iload1: H = LV;

iload2: MAR = H + MBRU; rd;

iload3: MAR = SP = SP + 1;

iload4: PC = PC + 1; fetch; wr;

iload5: TOS = MDR; goto Main1;

ISTORE *varnum*

istore1: H = LV;

istore2: MAR = H + MBRU;

istore3: MDR = TOS; wr;

istore4: MAR = SP = SP – 1; rd;

istore5: PC = PC + 1; fetch;

istore6: TOS = MDR; goto Main1;

WIDE

wide1: PC = PC + 1; fetch; goto(x100 **or** MBR);

WIDE ILOAD *varnum*

wideload1: PC = PC + 1; fetch;

wideload2: H = MBR << 8;

wideload3: H = H or MBRU;

wideload4: MAR = H + LV; rd; goto iload3;

WIDE ISTORE *varnum*

widestore1: PC = PC + 1; fetch;

widestore2: H = MBR << 8;

widestore3: H = H or MBRU;

widestore4: MAR = H + LV; goto istore3;

LDC\_W *index*

ldc\_w1: PC = PC + 1; fetch;

ldc\_w2: H = MBRU << 8;

ldc\_w3: H = H or MBR;

ldc\_w4: MAR = H + CPP; rd; goto iload3;

IINC *varnum const*

iinc1: H = LV;

iinc2: MAR = H + MBRU; rd;

iinc3: PC = PC + 1; fetch;

iinc4: H = MDR;

iinc5: PC = PC + 1; fetch;

iinc6: MDR = H + MBR; wr; goto Main1;

GOTO *offset*

goto1: OPC = PC – 1;

goto2: PC = PC + 1; fetch;

goto3: H = MBR << 8;

goto4: H = H or MBRU;

goto5: PC = H + OPC; fetch;

goto6: goto Main1;

IFLT offset (IFEQ similarly)

iflt1: MAR = SP = SP – 1; rd;

iflt2: OPC = TOS;

iflt3: TOS = MDR;

iflt4: N = OPC; if(N) goto T else goto F;

T: OPC = PC – 1; goto goto2;

F: PC = PC + 1;

F2: PC = PC + 1; fetch;

F3: goto Main1;

IF\_CMPEQ offset

ifcmpeq1: MAR = SP = SP – 1; rd;

ifcmpeq2: MAR = SP = SP – 1;

ifcmpeq3: H = MDR; rd;

ifcmpeq4: OPC = TOS;

ifcmpeq5: TOS = MDR;

ifcmpeq6: Z = OPC – H; if Z goto T else goto F;

INVOKEVIRTUAL disp

invoke1: PC = PC + 1; fetch;

invoke2: H = MBRU << 8;

invoke3: H = H or MBRU;

invoke4: MAR = H + CPP; rd;

invoke5: OPC = PC + 1;

invoke6: PC = MDR; fetch;

invoke7: PC = PC + 1; fetch;

invoke8: H = MBRU << 8;

invoke9: H = H of MBRU;

invoke10: PC = PC + 1; fetch

invoke11: TOS = SP - H

invoke12: MAR = TOS = TOS + 1;

invoke13: PC = PC + 1; fetch;

invoke14: H = MBRU << 8;

invoke15: H = H or MBRU;

invoke16: MDR = SP + H + 1; wr;

invoke17: MAR = SP = MDR;

invoke18: MDR = OPS; wr;

invoke19: MAR = SP = SP + 1;

invoke20: MDR = LV; wr;

invoke21: PC = PC + 1; fetch;

invoke22: LV = TOS; goto Main1;

IRETURN

ireturn1: MAR = SP = LV; rd;

ireturn2:

ireturn3: MAR = LV = MDR; rd;

ireturn4: MAR = LV + 1;

ireturn5: PC = MDR; rd; fetch;

ireturn6: MAR = SP;

ireturn7: LV = MDR;

ireturn8: MDR = TOS; wr; goto Main1;