1) ADDinstr:

```
EXPRSTACK[--ESP-1] += EXPRSTACK[ESP];
```

2) ANDinstr extends ZeroOperandInstruction:

```
EXPRSTACK[--ESP - 1] = EXPRSTACK[ESP - 1] & EXPRSTACK[ESP];
```

3) CALLSTATMETHODinstr extends OneOperandInstruction:

```
TJ.data[(ASP - POINTERTAG)] = PC;
ASP++;
PC = this.operand;
```

4) CHANGESIGNinstr extends ZeroOperandInstruction:

```
EXPRSTACK[ESP - 1] *= -1;
```

5) DIVinstr extends ZeroOperandInstruction:

```
EXPRSTACK[--ESP - 1] = EXPRSTACK[ESP - 1] / EXPRSTACK[ESP];
```

6) EQinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[--ESP - 1] == EXPRSTACK[ESP]) {
    EXPRSTACK[ESP - 1] = 1;
 } else {
    EXPRSTACK[ESP - 1] = 0;
}
```

7) GEinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[--ESP - 1] >= EXPRSTACK[ESP]) {
        EXPRSTACK[ESP - 1] = 1;
} else {
        EXPRSTACK[ESP - 1] = 0;
}
```

8) GTinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[--ESP - 1] > EXPRSTACK[ESP]) {
        EXPRSTACK[ESP - 1] = 1;
} else {
        EXPRSTACK[ESP-1] = 0;
}
```

9) INITSTKFRMinstr extends OneOperandInstruction:

```
TJ.data[ASP - POINTERTAG] = FP;
FP = ASP++;
ASP = ASP + this.operand;
```

10) JUMPinstr extends OneOperandInstruction:

```
PC = this.operand;
```

11) JUMPONFALSEinstr extends OneOperandInstruction:

```
if(EXPRSTACK[--ESP] == 0) {
        PC = this.operand;
}
```

12) LEinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[--ESP - 1] <= EXPRSTACK[ESP]) {
        EXPRSTACK[ESP - 1] = 1;
} else {
        EXPRSTACK[ESP - 1] = 0;
}
```

13) LOADFROMADDRinstr extends ZeroOperandInstruction:

```
int a = EXPRSTACK[ESP - 1];
EXPRSTACK[ESP - 1] = TJ.data[a - POINTERTAG];
```

14) LTinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[--ESP - 1] < EXPRSTACK[ESP]) {
        EXPRSTACK[ESP - 1] = 1;
} else {
        EXPRSTACK[ESP - 1] = 0;
}
```

15) MODinstr extends ZeroOperandInstruction:

```
EXPRSTACK[--ESP - 1] = EXPRSTACK[ESP - 1] % EXPRSTACK[ESP];
```

16) MULinstr extends ZeroOperandInstruction:

```
EXPRSTACK[--ESP - 1] = EXPRSTACK[ESP - 1] * EXPRSTACK[ESP];
```

17) NEinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[--ESP - 1] != EXPRSTACK[ESP]) {
        EXPRSTACK[ESP - 1] = 1;
} else {
        EXPRSTACK[ESP - 1] = 0;
}
```

18) NOTinstr extends ZeroOperandInstruction:

```
if(EXPRSTACK[ESP - 1] == 0) {
        EXPRSTACK[ESP - 1] = 1;
 } else {
        EXPRSTACK[ESP - 1] = 0;
 }
```

19) ORinstr extends ZeroOperandInstruction:

```
EXPRSTACK[--ESP - 1] = EXPRSTACK[ESP - 1] | EXPRSTACK[ESP];
```

20) PASSPARAMinstr extends ZeroOperandInstruction:

```
TJ.data[ASP++ - POINTERTAG] = EXPRSTACK[--ESP];
```

21) PUSHLOCADDRinstr extends OneOperandInstruction:

```
EXPRSTACK[ESP++] = FP + this.operand;
```

22) PUSHNUMinstr extends OneOperandInstruction:

```
 EXPRSTACK[ESP++] = this.operand;
```

23) PUSHSTATADDRinstr extends OneOperandInstruction:

```
EXPRSTACK[ESP++] = this.operand + POINTERTAG;
```

24) RETURNinstr extends OneOperandInstruction:

```
ASP = FP;
FP = TJ.data[(ASP--) - POINTERTAG];
PC = TJ.data[ASP - POINTERTAG];
ASP -= this.operand;
```

25) SAVETOADDRinstr extends ZeroOperandInstruction:

```
int a = EXPRSTACK[ESP - 2] - POINTERTAG;
TJ.data[a] = EXPRSTACK[ESP - 1];
ESP = ESP - 2;
```

26) SUBinstr extends ZeroOperandInstruction:

```
EXPRSTACK[--ESP - 1] = EXPRSTACK[ESP - 1] - EXPRSTACK[ESP];
```

27) WRITEINTinstr extends ZeroOperandInstruction:

```
System.out.print(EXPRSTACK[--ESP]);
```

28) WRITELNOPinstr extends ZeroOperandInstruction:

```
System.out.println();
```

29) WRITESTRINGinstr extends TwoOperandInstruction:

```
for(int a = this.firstOperand; a <= this.secondOperand; a++) {
    System.out.print((char)TJ.data[a]);
}
```