Computer Organization
CS 231-01

Conversion to IJVM Bytecode Part 2

Dr. William H. Robinson
October 27, 2004

http://eecs.vanderbilt.edu/courses/cs231/

Topics

“If you can meet with triumph and disaster, and treat those two imposters just the same.”
– Rudyard Kipling

• Administrative stuff
  – Programming teams
  – Mic1 Simulator

• Java bytecode

• IJVM instruction set

Programming Teams

• Weblink from class schedule page

• Team 1: Noor, Ted, Tamim
• Team 2: Michael, David F., Nikhil
• Team 3: Theodor, Brandon, Jon
• Team 4: Brett, Jeb, John W.
• Team 5: Dante', Clay, William
• Team 6: John L., Lauren, David R.

IJVM Instruction Set

<table>
<thead>
<tr>
<th>Hex</th>
<th>Mnemonic</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>BIPUSH byte</td>
<td>Push byte onto stack</td>
</tr>
<tr>
<td>0x59</td>
<td>DUP</td>
<td>Copy top word on stack and push onto stack</td>
</tr>
<tr>
<td>0x47</td>
<td>GOTO offset</td>
<td>Unconditional branch</td>
</tr>
<tr>
<td>0x93</td>
<td>IADD</td>
<td>Pop two words from stack, push their sum</td>
</tr>
<tr>
<td>0x7E</td>
<td>IAND</td>
<td>Pop two words from stack, push Boolean AND</td>
</tr>
<tr>
<td>0x99</td>
<td>IFEQ offset</td>
<td>Pop word from stack and branch if it is zero</td>
</tr>
<tr>
<td>0x0B</td>
<td>IFLT offset</td>
<td>Pop word from stack and branch if it is less than zero</td>
</tr>
<tr>
<td>0x9F</td>
<td>IFS_EQ offset</td>
<td>Pop two words from stack, branch if equals</td>
</tr>
<tr>
<td>0x84</td>
<td>IINC vartype</td>
<td>Add a constant to a local variable</td>
</tr>
<tr>
<td>0x15</td>
<td>ILOAD vartype</td>
<td>Push local variable onto stack</td>
</tr>
<tr>
<td>0x83</td>
<td>INVOKEVIRTUAL</td>
<td>Invoke a method</td>
</tr>
<tr>
<td>0x80</td>
<td>IOP</td>
<td>Pop two words from stack, push Boolean OR</td>
</tr>
<tr>
<td>0x4C</td>
<td>RETURN</td>
<td>Return from method with integer value</td>
</tr>
<tr>
<td>0x36</td>
<td>ISTORE vartype</td>
<td>Pop word from stack and store in local variable</td>
</tr>
<tr>
<td>0x54</td>
<td>ISUB</td>
<td>Pop two words from stack, push their difference</td>
</tr>
<tr>
<td>0x33</td>
<td>LDC_V1_INDEX</td>
<td>Push constant from constant pool onto stack</td>
</tr>
<tr>
<td>0x00</td>
<td>NOP</td>
<td>Do nothing</td>
</tr>
<tr>
<td>0x77</td>
<td>POP</td>
<td>Delete word on top of stack</td>
</tr>
<tr>
<td>0x5F</td>
<td>SWAP</td>
<td>Swap the two top words on the stack</td>
</tr>
<tr>
<td>0x4C</td>
<td>WIDE</td>
<td>Prefix instructions; next instruction has a 16-bit index</td>
</tr>
</tbody>
</table>

Figure 4.11. The IJVM instruction set. The operands byte, vartype, and vartype are 1 byte. The operands index, index, and
otherwise are 2 bytes.
IJVM Assembly Mnemonics

- **ISTORE variable**
  - Pops a value from the stack and places it in the variable
- **ILOAD variable**
  - Pushes the value in the variable onto the stack
- **In Java bytecode**
  - variable is represented by varnum — an 8-bit value
- **WIDE**
  - Prefix instruction
  - Indicates the next instruction has a 16-bit index
  - WIDE ISTORE or WIDE ILOAD

“Simple” Addition

// i = 12; j = 4; n = i + j;
BIPUSH 12
ISTORE i
BIPUSH 4
ISTORE j
ILOAD i
ILOAD j
IADD
ISTORE n
HALT

“Simple” Addition Bytecode

// i = 12; j = 4; n = i + j;
BIPUSH 12 0x10 0x0C
ISTORE i 0x36 0x00
BIPUSH 4 0x10 0x04
ISTORE j 0x36 0x01
ILOAD i 0x15 0x00
ILOAD j 0x15 0x01
IADD 0x60
ISTORE n 0x36 0x02
HALT 0xFF

IJVM Instruction Set

- **Mnemonics** — 1 byte
- **The operands byte, const, varnum** — 1 byte
- **The operands disp, index, offset** — 2 bytes
Using Offset

• **Operand of all branch instructions**
  - GOTO, IFEQ, IFLT, IF_ICMPEQ

• **Changes the value of PC**
  - 16-bit signed value
    • Use 2’s complement notation
    • First byte fetched from method area is the “high” byte
    • Second byte fetched from method area is the “low” byte
  - Added to the PC value at the start of the instruction
    • Not the value after fetching the 2 offset bytes

IJVM Assembly Mnemonics

• **IINC variable value**
  - Adds the value to the variable and stores the result back in the variable

• **DUP**
  - Copies the value on the top on the stack and pushes the same value onto the stack
  - The value at the top of the stack and the value one below the top of the stack will be the same (duplicated).

Other Conditional Statements

• **IFLT label name**
  - Pop a value off the stack
  - If the value is less than zero jump to the label

• **IF_ICMPEQ label name**
  - Pop 2 values off the stack
  - If they are equal jump to the label.

Labels and Conditional Statements

• **Label name: A reference to a place in the program**
  
  L1: BIPUSH 8

• **IFEQ label name**
  IFEQ L1
  - Pops a value off the stack if it is equal to zero than the program jumps to the label.

• **GOTO label name**
  GOTO L1
  - Unconditional jump to the label
### Loops in Machine Language

#### High-level language:
```
for (int i = 3; i >= 0; i--)
```

#### Bytecode
```
0x10 0x00 0xff 0xa7 0xff 0xf8 0x36 0x03
```

### Summary
- The method area of the IJVM memory model holds the Java bytecode.
- There is a 1:1 mapping between ISA instructions and assembly instructions.
- Offsets are added to the PC value to redirect program flow to the corresponding label in the assembly code.