

|                      |   |
|----------------------|---|
| add $r_2, r_1$       | $r_1 \leftarrow (r_1) + (r_2)$  |
| addf $r_2, r_1$      | $r_1 \leftarrow (r_1) + (r_2)$ (floating)   |
| alc $r_2, r_1$       | $r_1 \leftarrow$ address of $(r_2)$ words allocated on heap                             |
| and $r_2, r_1$       | $r_1 \leftarrow (r_1) \& (r_2)$   |
| bra l                | pc $\leftarrow$ l   |
| brf $r_1, l$         | if $\sim(r_1)$ then pc $\leftarrow$ l   |
| cal d, $r_1$         | call method at address $(r_1)$ with difference d between calling level and called level |
| cls c, s             | class record with label c and superclass s  |
| cst v, $r_1$         | $r_1 \leftarrow v$  |
| cstf v, $r_1$        | $r_1 \leftarrow v$ (floating)   |
| div $r_2, r_1$       | $r_1 \leftarrow (r_1) \text{ DIV } (r_2)$   |
| divf $r_2, r_1$      | $r_1 \leftarrow (r_1) \div (r_2)$ (floating)  |
| eq l $r_2, r_1$      | $r_1 \leftarrow (r_1) = (r_2)$  |
| eqlf $r_2, r_1$      | $r_1 \leftarrow (r_1) = (r_2)$ (floating)   |
| geq $r_2, r_1$       | $r_1 \leftarrow (r_1) \geq (r_2)$   |
| geqf $r_2, r_1$      | $r_1 \leftarrow (r_1) \geq (r_2)$ (floating)  |
| gtr $r_2, r_1$       | $r_1 \leftarrow (r_1) > (r_2)$  |
| gtrf $r_2, r_1$      | $r_1 \leftarrow (r_1) > (r_2)$ (floating)   |
| leq $r_2, r_1$       | $r_1 \leftarrow (r_1) \leq (r_2)$   |
| leqf $r_2, r_1$      | $r_1 \leftarrow (r_1) \leq (r_2)$ (floating)  |
| lt n $r_2, r_1$      | $r_1 \leftarrow (r_1) < (r_2)$  |
| ltnf $r_2, r_1$      | $r_1 \leftarrow (r_1) < (r_2)$ (floating)   |
| mth m                | class record entry for method with entry point m  |
| mul $r_2, r_1$       | $r_1 \leftarrow (r_1) \times (r_2)$   |
| mulf $r_2, r_1$      | $r_1 \leftarrow (r_1) \times (r_2)$ (floating)  |
| ndx $r_2, l, s, r_1$ | $r_1 \leftarrow (r_1) + [(r_2) - 1] * s$  |
| neg $r_1$            | $r_1 \leftarrow -(r_1)$   |
| negf $r_1$           | $r_1 \leftarrow -(r_1)$ (floating)  |
| neq $r_2, r_1$       | $r_1 \leftarrow (r_1) \neq (r_2)$   |
| neqf $r_2, r_1$      | $r_1 \leftarrow (r_1) \neq (r_2)$ (floating)  |
| not $r_1$            | $r_1 \leftarrow \sim(r_1)$  |
| ntr p, s             | establish entry point for method with label p and s words of local storage on stack     |
| or $r_2, r_1$        | $r_1 \leftarrow (r_1) \mid (r_2)$   |
| psh $r_1$            | (sp) $\leftarrow (r_1)$ , sp $\leftarrow$ (sp) + 1                                      |
| rem $r_2, r_1$       | $r_1 \leftarrow (r_1) \text{ REM } (r_2)$   |

|  |  |
|--|--|
| <code>rtn s</code>                             | return from method which has <code>s</code> words of parameter storage   |
| <code>rtn r<sub>1</sub>, s</code>              | return from function method which has <code>s</code> words of parameter storage and result in register <code>r<sub>1</sub></code> , <code>rr ← (r<sub>1</sub>)</code>  |
| <code>sel d, r<sub>1</sub></code>              | <code>r<sub>1</sub> ← (r<sub>1</sub>) + d</code>   |
| <code>sto r<sub>2</sub>, r<sub>1</sub></code>  | <code>(r<sub>1</sub>) ← (r<sub>2</sub>)</code>   |
| <code>str l, s</code>                          | storage for string literal <code>s</code> (enclosed in <code>" "</code> ), with label <code>l</code>   |
| <code>sub r<sub>2</sub>, r<sub>1</sub></code>  | <code>r<sub>1</sub> ← (r<sub>1</sub>) - (r<sub>2</sub>)</code>   |
| <code>subf r<sub>2</sub>, r<sub>1</sub></code> | <code>r<sub>1</sub> ← (r<sub>1</sub>) - (r<sub>2</sub>) (floating)</code>  |
| <code>val r<sub>1</sub></code>                 | <code>r<sub>1</sub> ← ((r<sub>1</sub>))</code>   |
| <code>var d, o, r<sub>1</sub></code>           | <code>r<sub>1</sub> ← address of variable with reference pair (d, o)</code>  |
| <code>gbln r<sub>1</sub></code>                | <code>r<sub>1</sub> ← next boolean in input</code>   |
| <code>gchr r<sub>1</sub></code>                | <code>r<sub>1</sub> ← next character in input</code>   |
| <code>gflt r<sub>1</sub></code>                | <code>r<sub>1</sub> ← next float in input</code>   |
| <code>gint r<sub>1</sub></code>                | <code>r<sub>1</sub> ← next integer in input</code>   |
| <code>gstr r<sub>1</sub></code>                | <code>r<sub>1</sub> ← address of next string in input</code>   |
| <code>pbln r<sub>1</sub></code>                | <code>output ← (r<sub>1</sub>) as boolean</code>   |
| <code>pchr r<sub>1</sub></code>                | <code>output ← (r<sub>1</sub>) as character</code>   |
| <code>pflt r<sub>1</sub></code>                | <code>output ← (r<sub>1</sub>) as float</code>   |
| <code>pint r<sub>1</sub></code>                | <code>output ← (r<sub>1</sub>) as integer</code>   |
| <code>pstr r<sub>1</sub></code>                | <code>output ← string at (r<sub>1</sub>)</code>  |
| <code>pln</code>                               | newline on output  |
| <code>end l, c, m</code>                       | create driver routine, initialize I/O, create object record with <code>l</code> words of storage and class record pointer <code>c</code> , establish AR stack and call constructor with entry point <code>m</code> |

## Notes

1. General purpose registers are `r0` through `r13`
2. Special purpose registers are `pc` (program counter), `sp` (stack pointer), `fp` (frame pointer), `rr` (function return register)
3. Only single precision floating point (`float`) operations are supported.
4. `boolean` constants `true` and `false` are supported as bit (`integer`) values 1 and 0, respectively.