Generic Assembly Language (GAL) Quick Reference

Line format:

```
label   operator   operands        // Comments follow "//" anywhere on a line
```

label: alpha numeric string beginning with a letter (begins as first character of line)
A "label" can occur within either instructions or data definitions.
Every program must have a label "main" which is where execution will begin.

Data definitions:

```
label   int        value           // allocate memory location "label" and set to "value"
```

Instructions:

operands:

```
                   Rn              // the name of a general register (R1 -> R15)
                   Ri              // the name of a general register (R1 -> R15)
                   mem             // the label (or name) of a memory location
                   value           // a constant value
```

operators:

```
load       Rn, mem          // load contents of "mem" into Rn
load       Rn, mem, Ri      // load the contents of memory location
                            // "mem + (contents of Ri)" into Rn
store      Rn, mem          // store contents of Rn into memory location "mem"
store      Rn, mem, Ri      // store the contents of Rn into the memory
                            // location "mem + (contents of Ri)"
loadK      Rn, value        // put the constant "value" into Rn
copy       Rn, Ri           // copy the contents of Ri into Rn
add        Rn, Ri           // add contents of Rn and Ri
                            // the result goes into Rn
sub        Rn, Ri           // subtract the contents of register Ri from the
                            // contents of Rn, the result goes in Rn
mult       Rn, Ri           // multiply contents of registers Rn and Ri
                            // the result goes into Rn
div        Rn, Ri           // divide the contents of register Rn by the contents of Ri
                            // integer quotient goes in Rn and remainder goes into Rn+1
jmp        mem              // jump to location mem
jmpZero    Rn,mem           // if Rn contains the value 0, jump to location mem
jmpNZero   Rn,mem           // if Rn does not contain  0, jump to location mem
jmpPos     Rn,mem           // if Rn is positive, jump to location mem
jmpNeg     Rn,mem           // if Rn is negative, jump to location mem
return                      // end execution and return to operating system
```

Input/Output Instructions:

```
readChar   mem              // Read 1 character into memory location "mem"
writeChar  mem              // Write 1 character from memory location "mem"
printReg   Rn               // Debug tool: print information about register Rn
printMsg   "message"        // Debug tool: print a message string (can contain spaces)
```