```
    ,-'~~~'-,
  .~      `. ~.
 /    8     |  \
:         ,'    :
|     .--~      |
!    ;          !
 \  |    8     /
  `. ',      .'
    `-.___.-`

  B A L A N C E
  User's Manual
```

I. Introduction

Night and day. Beauty and truth. Oxygen and phlogiston. Everywhere we look there are perfect opposites. Balance is a programming language based on the concept of harmoniously coexisting duals. In this language, every operation has an equal and opposite reaction. This ensures that the machine state does not stray from equilibrium. Because it performs two operations for each instruction, the language is also twice as fast as single-operation languages.

Balance programs are run inside an 8-bit machine with the following features:

* CODE: an arbitrarily long immutable stream of bytes
* M[0..255]: 256 8-bit bytes of memory
* IP: the instruction pointer (an arbitrary non-negative value)
* IS: the instruction speed (ranging from -16 to +15)
* sR[0..3]: four 8-bit source registers
* dR[0..1]: two 8-bit destination registers
* four instructions

Each instruction is specified by a single 8-bit byte. The bits are numbered from most to least meaningful as follows:

```
         lmb
   .--------.
   |76543210|
   `--------'
  mmb
```

* The bits 7,6,5 specify the opcode

and depending on the instruction:

* bits 4,3,2,1,0 specify an immediate value IMM

or

* bit 4 denotes a destination register D
* bits 3,2 denote the first source register S1
* bits 1,0 denote the second source register S2

Every problem in computer science can be solved by an additional layer of indirection. Balance thus provides this facility automatically: Each of the source and destination registers is an indirect register. For instance, most instructions do not work on the contents of registers S1 and S2 directly, but use the contents of S1 and S2 as indices into the memory M. The result is not stored in D, but rather stored in the memory location indicated by the current contents of D.

The machine begins with IP = 0 and IS = 1. At each step of the machine, an instruction is fetched from CODE[IP] (where CODE[0] is the

first instruction, CODE[1] the second, etc.). The instruction is executed, and then IP is increased (or decreased) by the value of IS, modulo the length of CODE.

II. Instruction Reference

The four instructions of Balance are MATH, LOGIC, SCIENCE, and PHYSICS. The following is their specification; some examples are given in a separate section below.

```
Opcode  (bits)   Description
 MATH    001
```

MATH performs addition and its dual, subtraction. These act on different registers so that the math is not undone. All operations are modular with respect to the number of relevant bits. Source registers are represented with two bits, so if S1 is 3, then sR[S1+1] is sR[0]. Similarly, dR[1+1] is dR[0]. Quantities in memory are eight bits, so 250 + 20 is 14.

```
M[ dR[D+1] ] <- M[ sR[S1+1] ]  -  M[ sR[S2+1] ]
M[ dR[D]   ] <- M[ sR[S1]   ]  +  M[ sR[S2]   ]
```

```
 LOGIC   010
```

LOGIC performs bitwise 'and' as well as its perfect dual, bitwise 'exclusive or.'

```
M[ dR[D+1] ] <- M[ sR[S1+1] ]  XOR  M[ sR[S2+1] ]
M[ dR[D]   ] <- M[ sR[S1]   ]  AND  M[ sR[S2]   ]
```

```
 SCIENCE 000
```

SCIENCE tests a hypothesis and determines the speed at which the program progresses. When executed, it sets the instruction speed IS to immediate value IMM, as long as the memory cell indicated by sR[0] does not contain 0. Because this instruction behaves specially when the memory cell contains 0, it also behaves specially if IS is set to zero: the machine then halts. The value IMM is treated as a signed five-bit number in two's complement form, so it can take on values from -16 to +15.

```
if M[ sR[0] ] = 0 then (nothing)
otherwise IS <- IMM

if IS = 0 then HALT
else (nothing)
```

```
 PHYSICS 011
```

PHYSICS changes what the registers reference, in both a linear and angular way. The immediate value IMM, treated as a signed five-bit number, is added to the register sR[0] so that it may reference a different memory cell. The instruction also rotates the values between some subset of the registers, according to a bitmask derived from IMM. The source register sR[0] is always part of the rotated set, so the bitmask used is a 6 bit number where the lowest 5 bits are the same as IMM and the sixth bit is always 1.

```
                 sR[0] <- sR[0] + (IMM as signed 5-bit number)

                 let L=L0,...,L4 be the registers
                     dR[1], dR[0], sR[3], sR[2], sR[1]
                 then let C be the list of n elements Li
                     such that bit i is set in IMM
                     (bit 0 is the least significant,
                      bit 4 is the most significant)
                 then let Cs be the list (sR[0], C0, ..., C(n-1))
                 and  let Cd be the list (C0, ..., C(n-1), sR[0])
                 then, simultaneously
                      Cd0 <- Cs0
                      ...
                      Cdn <- Csn
```

Any other opcode stands for the instruction BAIL, which will cause the machine to terminate in failure. Programmers sometimes insert such bugs deliberately in order to quickly halt the interpreter during testing.

III. Examples

```
If M[sR[0]] = 0, IP = 3, IS = 6, length(CODE) = 100,
and CODE[IP] = SCIENCE 12,
then in the next cycle IP = 9 and IS = 6.

If M[sR[0]] = 9, IP = 3, IS = 6, length(CODE) = 100,
and CODE[IP] = SCIENCE 12,
then in the next cycle IP = 15 and IS = 12.

If sR = {0, 1, 2, 3}, dR = {4, 5}, M = {2, 3, 5, 7, 11, 13, 17, ...},
and CODE[IP] = MATH (0, 3, 1)
then in the next cycle M = {2, 3, 5, 7, 10, 253, 17, ...}.

If sR = {0, 1, 2, 3}, dR = {4, 5}, M = {2, 3, 5, 7, 11, 13, 17, ...},
and CODE[IP] = LOGIC (0, 3, 1)
then in the next cycle M = {2, 3, 5, 7, 3, 7, 17, ...}.

If sR = {0, 1, 2, 3}, dR = {4, 5}
and CODE[IP] = PHYSICS -1
then sR[0] is updated with sR[0] + -1 = 255
and in the next cycle sR = {1, 2, 3, 4}, dR = {5, 255}.

If sR = {0, 1, 2, 3}, dR = {4, 5}
and CODE[IP] = PHYSICS -16
Then sR[0] is updated with sR[0] + (-16) = -16.
The bitmask for rotation is   1  1  0  0   0  0
       for the register set {-16, 1, 2, 3} {4, 5},
so in the next cycle sR = {1, -16, 2, 3}, dR = {4, 5}.

If sR = {0, 1, 2, 3}, dR = {4, 5}
and CODE[IP] = PHYSICS 15
then sR[0] is updated with sR[0] + 15 = 15.
The bitmask for rotation is   1  0  1  1   1  1
       for the register set {15, 1, 2, 3} {4, 5}
so in the next cycle sR = {2, 1, 3, 4}, dR = {5, 15}.
```

IV. Syntax

The Balance language concrete syntax is simply a single line of bytes, each written as two hexadecimal digits, with no whitespace or other characters.

V. Balance Certified Professional Program (BCPP)

As a professional programmer, you are invited to join one of the industry's leading programmer certification programs. BCPP certification can be obtained automatically by solving a series of challenge problems and verifying the results using the supplied program "certify".

Please see the file PUZZLES for the list of challenge problems. To certify a solution, stored in a file called "solve.bal", to the problem called "prop", run the command

```
   certify prop solve.bal
```

Because a professional programmer knows that shorter code is better code, your certified skill level depends on the length of the program that you submit to the certifier.

A challenge problem consists of a description of the initial register and memory values and the desired final configuration. A program solves the challenge if it achieves the final configuration and halts gracefully (SCIENCE 0 with M[sR[0]] <> 0).

Accepted applicants will receive an engraved sandstone diploma and will be added to the 19101 electronic edition of "Who's Who in Computerology." Please allow 4-6 weeks for delivery.