Appendix C

Vole: A Simple Machine Language

In this appendix, we present a simple but representative machine language, named for the simple and hypothetical machine it would run on. We begin by explaining the architecture of the machine itself.

The Vole Architecture[1]

Let us say that our *Vole* computer has 16 general-purpose registers numbered 0x0 through 0xF. Each register is one byte (eight bits) long. For identifying registers within instructions, each register is assigned the unique four-bit pattern that represents its register number. Thus, register 0x0 is identified by binary 0000, and register 0x4 is identified by binary 0100.

There are 256 cells in the Vole's main memory. Each cell is assigned a unique address consisting of an integer in the range of 0 to 255. An address can therefore be represented by a pattern of eight bits ranging from 00000000 to 11111111 (or a hexadecimal value in the range of 0x00 to 0xFF).

Floating-point values are assumed to be stored in an eight-bit format discussed in Section 1.7 and summarized in Figure 1.24.

The Vole's Machine Language

Each Vole machine instruction is two bytes long. The first 4 bits provide the op-code; the last 12 bits make up the operand field. The table that follows lists the instructions in hexadecimal notation together with a short description of each. The letters R, S, and T are used in place of hexadecimal digits in those fields representing a register identifier that varies depending on the particular application of the instruction. The letters X and Y are used in lieu of hexadecimal digits in variable fields not representing a register.

[1] *Vole* refers to a family of stout, short-tailed rodent species found across much of the world. Like the processor described in this appendix, they are small but effective.

Op-code	Operand	Description
0x1	RXY	LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* 0x14A3 would cause the contents of the memory cell located at address 0xA3 to be placed in register 0x4.
0x2	RXY	LOAD the register R with the bit pattern XY. *Example:* 0x20A3 would cause the value 0xA3 to be placed in register 0.
0x3	RXY	STORE the bit pattern found in register R in the memory cell whose address is XY. *Example*: 0x35B1 would cause the contents of register 0x5 to be placed in the memory cell whose address is 0xB1.
0x4	0RS	MOVE the bit pattern found in register R to register S. *Example:* 0x40A4 would cause the contents of register 0xA to be copied into register 0x4.
0x5	RST	ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* 0x5726 would cause the binary values in registers 0x2 and 0x6 to be added and the sum placed in register 0x7.
0x6	RST	ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. *Example:* 0x634E would cause the values in registers 0x4 and 0xE to be added as floating-point values and the result to be placed in register 0x3.
0x7	RST	OR the bit patterns in registers S and T and place the result in register R. *Example:* 0x7CB4 would cause the result of ORing the contents of registers 0xB and 0x4 to be placed in register 0xC.
0x8	RST	AND the bit patterns in registers S and T and place the result in register R. *Example:* 0x8045 would cause the result of ANDing the contents of registers 0x4 and 0x5 to be placed in register 0x0.
0x9	RST	XOR the bit patterns in registers S and T and place the result in register R. *Example:* 0x95F3 would cause the result of XORing the contents of registers 0xF and 0x3 to be placed in register 0x5.
0xA	R0X	ROTATE the bit pattern in register R one bit to the right X times. Each time, place the bit that started at the low-order end at the high-order end. *Example:* 0xA403 would cause the contents of register 0x4 to be rotated 3 bits to the right in a circular fashion.

Op-code	Operand	Description
0xB	RXY	`JUMP` to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. (The jump is implemented by copying XY into the program counter during the execute phase.) *Example:* 0xB43C would first compare the contents of register 0x4 with the contents of register 0x0. If the two were equal, the pattern 0x3C would be placed in the program counter so that the next instruction executed would be the one located at that memory address. Otherwise, nothing would be done and program execution would continue in its normal sequence.
0xC	000	`HALT` execution. *Example:* 0xC000 would cause program execution to stop.