INSTRUCTION FORMATS

<table>
<thead>
<tr>
<th>. . .</th>
<th>. . .</th>
<th>. . .</th>
<th>. . .</th>
<th>. . .</th>
<th>. . .</th>
</tr>
</thead>
</table>
Format 1: | opcode | d     | s     | t     |
| Format 2: | opcode | d     | addr  |
(0-6, A-B)  | (7-9, C-F)  

ARITHMETIC and LOGICAL operations
1: add     R[d] <- R[s] + R[t]
2: subtract R[d] <- R[s] - R[t]
3: and     R[d] <- R[s] & R[t]
4: xor     R[d] <- R[s] ^ R[t]
5: shift left R[d] <- R[s] << R[t]
6: shift right R[d] <- R[s] >> R[t]

TRANSFER between registers and memory
7: load address R[d] <- addr
8: load       R[d] <- mem[addr]
9: store      mem[addr] <- R[d]
A: load indirect R[d] <- mem[R[t]]
B: store indirect mem[R[t]] <- R[d]

CONTROL
0: halt       halt
C: branch zero if (R[d] == 0) pc <- addr
D: branch positive if (R[d] > 0) pc <- addr
E: jump register pc <- R[d]
F: jump and link R[d] <- pc; pc <- addr

Register 0 always reads 0.
Loads from mem[FF] come from stdin.
Stores to mem[FF] go to stdout.
pc starts at 10

16-bit registers
16-bit memory locations
8-bit program counter