

Tiny Instruction Set

(16 Mnemonics, 37 Instructions)

Note that brackets mean “content of address” while non-brackets mean number-literal

Group	Instruction	Byte Code	Description
1. Logic	AND a b	2 Ops, 3 bytes: 0x00 [a] [b] 0x01 [a] b	$M[a] = M[a] \text{ bit-wise and } M[b]$
	OR a b	2 Ops, 3 bytes: 0x02 [a] [b] 0x03 [a] b	$M[a] = M[a] \text{ bit-wise or } M[b]$
	XOR a b	2 Ops, 3 bytes: 0x04 [a] [b] 0x05 [a] b	$M[a] = M[a] \text{ bit-wise xor } M[b]$
	NOT a	1 Op, 2 bytes: 0x06 [a]	$M[a] = \text{bit-wise not } M[a]$
2. Memory	MOV a b	2 Ops, 3 bytes: 0x07 [a] [b] 0x08 [a] b	$M[a] = M[b]$, or the literal-set $M[a] = b$
3. Math	RANDOM a	2 Ops, 2 bytes: 0x09 [a]	$M[a] = \text{random value (0 to 25; equal probability distribution)}$
	ADD a b	2 Ops, 3 bytes: 0x0a [a] [b] 0x0b [a] b	$M[a] = M[a] + b$; no overflow support
	SUB a b	2 Ops, 3 bytes: 0x0c [a] [b] 0x0d [a] b	$M[a] = M[a] - b$; no underflow support
4. Control	JMP x	2 Ops, 2 bytes: 0x0e [x] 0x0f x	Start executing instructions at index of value $M[a]$ (So given a is zero, and $M[0]$ is 10, we then execute instruction 10) or the literal a-value
	JZ x a	4 Ops, 3 bytes: 0x10 [x] [a] 0x11 [x] a 0x12 x [a] 0x13 x a	Start executing instructions at index x if $M[a] == 0$ (This is a nice short-hand version of)
	JEQ x a b	4 Ops, 4 bytes: 0x14 [x] [a] [b] 0x15 x [a] [b] 0x16 [x] [a] b 0x17 x [a] b	Jump to x or $M[x]$ if $M[a]$ is equal to $M[b]$ or if $M[a]$ is equal to the literal b.

	JLS x a b	4 Ops, 4 bytes: 0x18 [x] [a] [b] 0x19 x [a] [b] 0x1a [x] [a] b 0x1b x [a] b	Jump to x or M[x] if M[a] is less than M[b] or if M[a] is less than the literal b.
	JGT x a b	4 Ops, 4 bytes: 0x1c [x] [a] [b] 0x1d x [a] [b] 0x1e [x] [a] b 0x1f x [a] b	Jump to x or M[x] if M[a] is greater than M[b] or if M[a] is greater than the literal b.
	HALT	1 Op, 1 byte: 0xff	Halts the program / freeze flow of execution
5. Utilities	APRINT a DPRINT a	4 Ops, 2 byte: 0x20 [a] (as ASCII; aprint) 0x21 a (as ASCII) 0x22 [a] (as Decimal; dprint) 0x23 a (as Decimal)	Print the contents of M[a] in either ASCII (if using APRINT) or as decimal (if using DPRINT). Memory ref or literals are supported in both instructions.