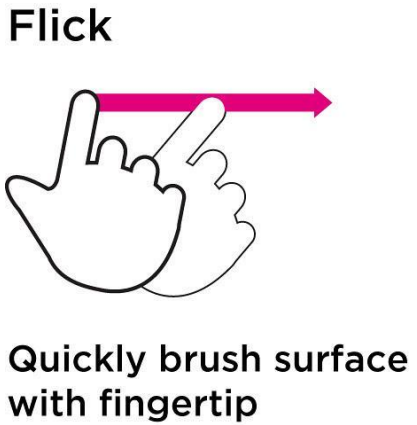
Bit-String FlickinggBit-String FlickinngBit-String FlickiingBit-String FlickkingBit-String Flic

Let’s flick it

Check out this bit string: **110010010101**

Ain’t that somethin’?

Let’s flick it!



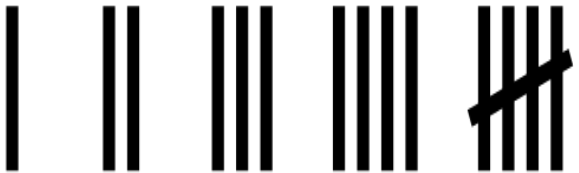
Unary vs. Binary Operations

Unary: an operation with only one operand

(e.g. negative symbol)

Binary: an operation with two operands

Do not change the length of the string



Bitwise Operations

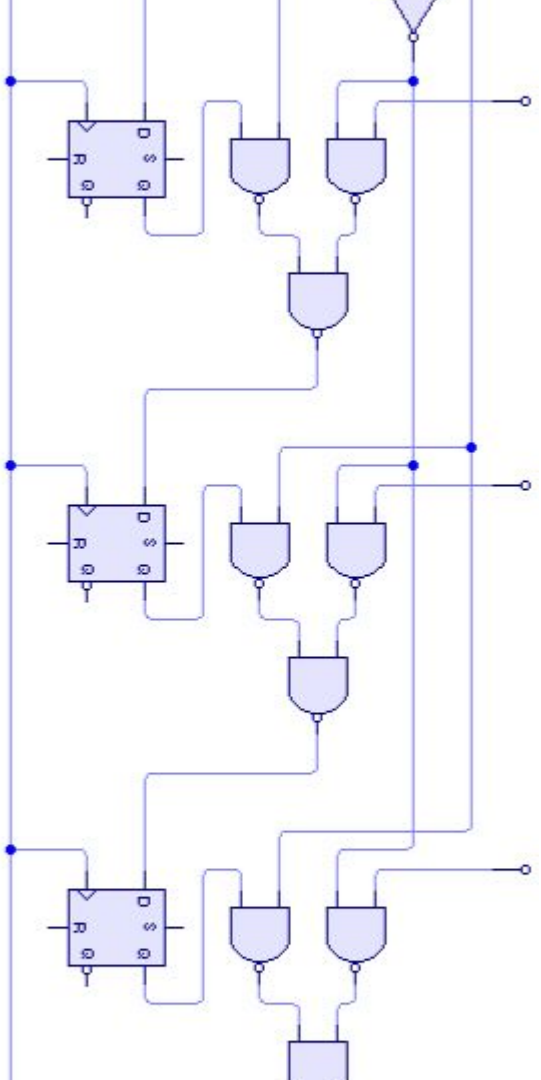
You are so...

Wise,
Witty, and
Wonderful.

- **NOT** (~ or ¬)
  - unary operation
  - negates each bit; flip from 0 to 1 or 1 to 0.
  - e.g. **NOT 10101 = 01010**
- **AND** (**&**)
  - binary operation
  - for each bit, value is 1 only if both operands have a 1 for that bit
  - e.g. **10101 AND 10001 = 10001**
- **OR** (**|**)
  - binary operation
  - for each bit, value is 1 if either operands have a 1 for that bit
  - e.g. **10101 OR 11001 = 11101**
- **XOR** (⊕)
  - "eXclusive or"
  - binary operation
  - for each bit, value is 1 if either operands have a 1 for that bit, but *not* both!
  - e.g. **10101 XOR 11001 = 01100**

Shift Operations

- **LSHIFT-*x* / RSHIFT-*x***
  - **left shift/right shift *x* times**
  - all bits move to the left/right
  - bits that move past the end are lost
  - zeros come in at the other end
  - e.g. **LSHIFT-2 10101 = 10100**
  - e.g. **RSHIFT-3 10101 = 00010**
- **LCIRC-*x* / RCIRC-*x***
  - **left circulate/right circulate *x* times**
  - all bits move to the left/right
  - bits that move past one end come out the other
  - e.g. **LCIRC-3 1101010 = 1010110**
  - e.g. **RCIRC-2 1101010 = 1011010**



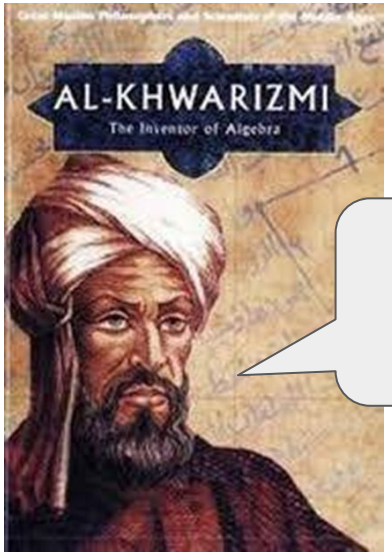
Order of Operations

NOT	Nice
SHIFT/CIRC	Sounding Cymbal
AND	And
XOR	Xylophone
OR	Orchestra

Bit Algebra, act i

List all possible values of x **(5 bits long)** that solve this equation:

LSHIFT-1 (10110 XOR (RCIRC-3 x) AND 11011) = 01100



What the flick is a bit?????

Bit Algebra, act ii

LSHIFT-1 (10110 XOR (RCIRC-3 x) AND 11011) = 01100

Let x, 5 bits long, equal **abcde.**

Plug in x: **RCIRC-3 abcde = cdeab**

(cdeab AND 11011) = cd0ab

(10110 XOR cd0ab) = Cd1Ab (Uppercase variables are the NOT value of the lowercase; XOR acts similar to NOT)

(LSHIFT-1 Cd1Ab) = d1Ab0

d1Ab0 = 01100

Bit Algebra, act iii

d1Ab0 = 01100

Therefore, **d = 0, A = 1 (so a = 0), and b = 0.**

Since c and e disappeared, they can have either value.

abcde = 00*0*

List all possible values of x:

00000, 00001, 00100, 00101

Make it easier, chapter 1

RCIRC-14 (LCIRC-23 01101) | (LSHIFT-1 10011) & (RSHIFT-2 10111)

Starting on the left:

You can combine the CIRCs into one: **23 left, 14 right = 9 left**

LCIRC-9 01101

Since the string is 5 bits long, shifting to the left or right 5 bits won't change anything, i.e. LCIRC-5 01101 = 01101

So we can subtract away multiples of 5 to make things easier:

LCIRC-9 01101 = LCIRC-4 01101 = 10110

Make it easier, chapter 2

10110 | (LSHIFT-1 10011) & (RSHIFT-2 10111)

Remember that AND comes first in the order of operations!

LSHIFT-1 10011 = 00110

RSHIFT-2 10111 = 00101

00110 & 00101 = 00100

10110 | 00100 = 10110

Yay, 10110

Do it: Q

```
101110 AND NOT 110110 OR (LSHIFT-3 101010)
```

Do it: A

101110 AND NOT 110110 OR (LSHIFT-3 101010)

101110 AND **001001** OR (LSHIFT-3 101010)

001000 OR (LSHIFT-3 101010)

001000 OR **010000**

011000

Do it: Q

```
RSHIFT-1 (LCIRC-4 (RCIRC-2 01101))
```

Do it: **A**

RSHIFT-1 (LCIRC-4 (RCIRC-2 01101))

RSHIFT-1 (LCIRC-4 **01011**)

RSHIFT-1 (**10101**)

01010