Multiplication (efficient)

```
      100100     x
 x     10110     y
 ------------
     100100
    100100
+100100
-----------
```

```
     527
 x    36
 -------
 . . . . . .  ← 6 × 527
 - - - - . . 0   30 × 527
```

Pseudocode to mul(x, y):

```
res = 0
for i in 0..15:
    if bit i of y is 1:
        res = res + x
    x = x + x          (in a real CPU, shift x left 1 bit)
```

2 ways:
① shift y right 1 bit every loop, test last bit
✗ Can't do on Hack CPU.
② Test w/ a mask.

To test bit i :

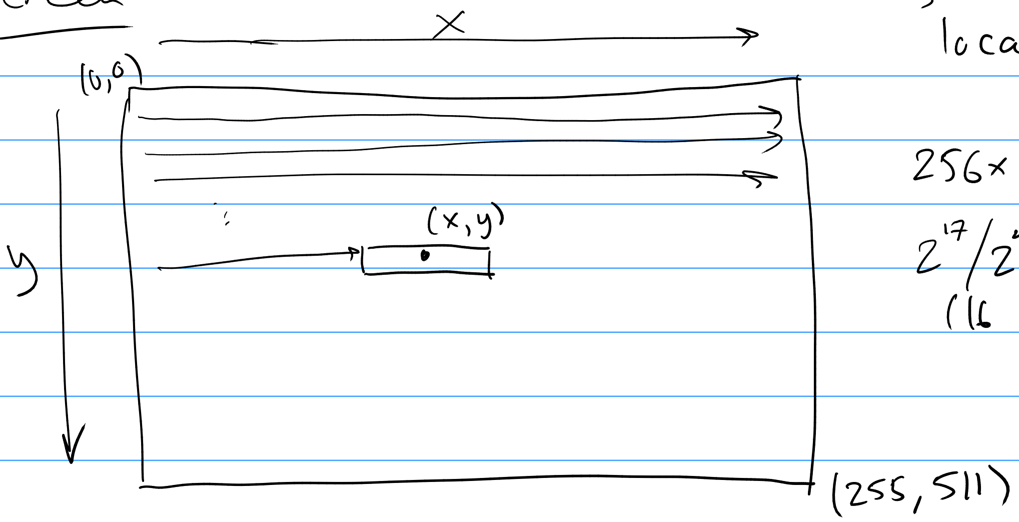
```
  10101b0110      ?
& 0000010000   ← bit mask, = 2^b where b is bit to test.
  ----------        keep track in a loop
  00000b0000        or precompute in array.
               ← See if this = 0.
```

Screen

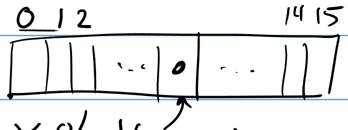
Screen starts at memory location $2^{14} = 16384$.

$256 \times 512 = 2^8 \cdot 2^9 = 2^{17}$ pixels.

$2^{17}/2^4 = 2^{13}$ memory locations
(16 pixels per memory loc.)



Pixel (x,y) is the $(512y + x)^{th}$ pixel.
Thus it is stored in memory location $(512y + x)/16$
$= 32y + x/16$ (offset from 2^{14}).



At bit $x \% 16$. ie. $x \& 1111$

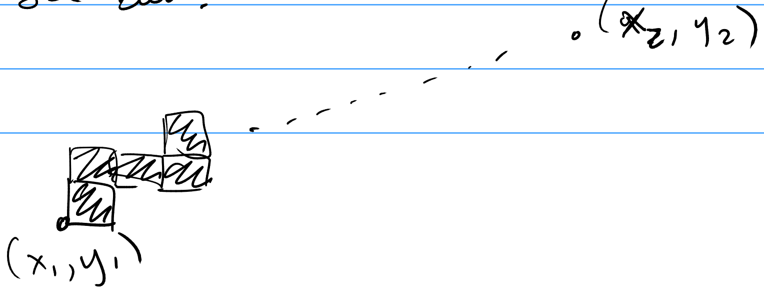
To set pixel (x,y):

- Get value @ SCREEN + $32y + x/16$
- Set bit $x \% 16$
- Put updated value back.

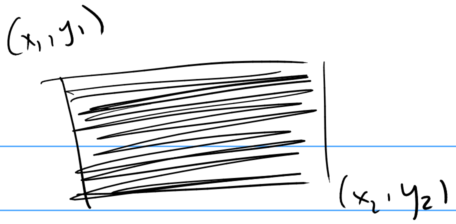
To set bit i:

- Use mask $0000\overset{i}{1}0000 = 2^i$
- To set bit i to 1, $v = v \,|\, 2^i$
- To clear bit i (set to 0), $v = v \,\&\, \underset{\substack{\downarrow \\ 1111101111}}{(\sim(2^i))}$

drawLine: Bresenham's algorithm.
See book!



drawRectangle



Loop + draw horizontal lines.

↳ drawLine

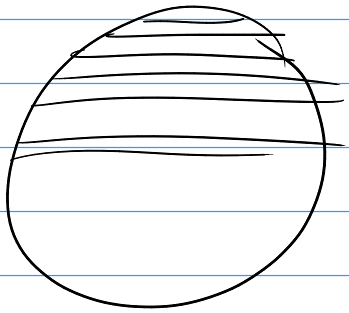
OR helper drawHorizLine

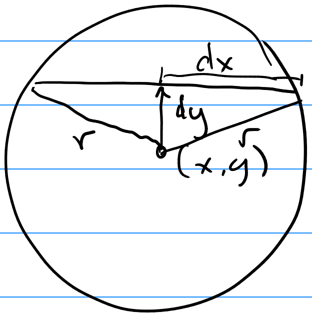
↳ pixel by pixel

OR 16 pixels @ once.

drawCircle

fill in by horiz. rows





$$dx^2 + dy^2 = r^2$$

$$\rightarrow dx = \sqrt{r^2 - dy^2}$$