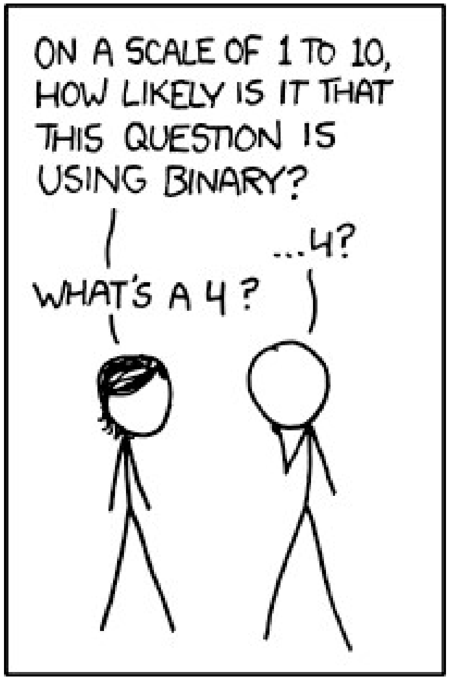
CS 101 Today...

Our top-10 list of binary jokes...

There are only 10 types of people in the world: Those who understand binary and those who don't.


ON A SCALE OF 1 TO 10, HOW LIKELY IS IT THAT THIS QUESTION IS USING BINARY?
WHAT'S A 4 ?
...4?

Looking Back

Looking Forward

Computing as composition

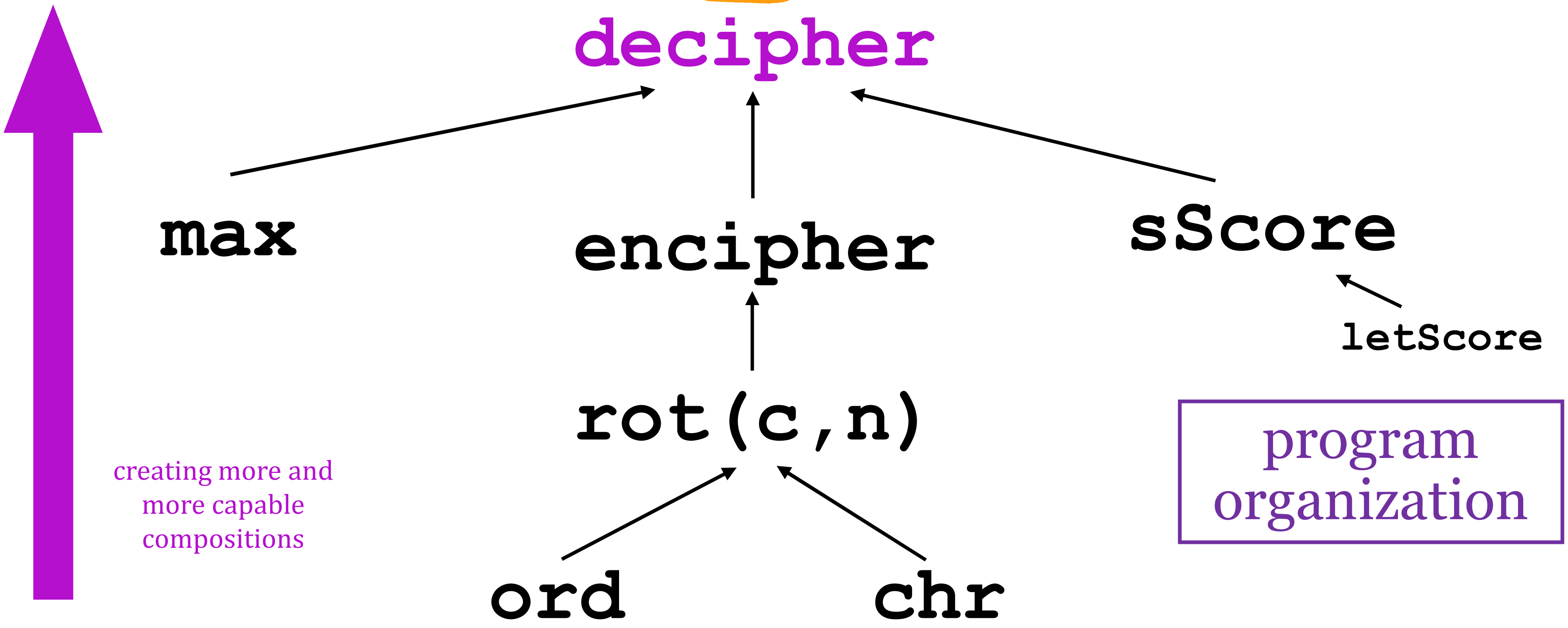
clay **== functions**

Computing as representation

clay **== data & *bits***

Some legs to stand on... ?

This is heady stuff!



decipher

max

encipher

sScore

letScore

rot(c,n)

ord

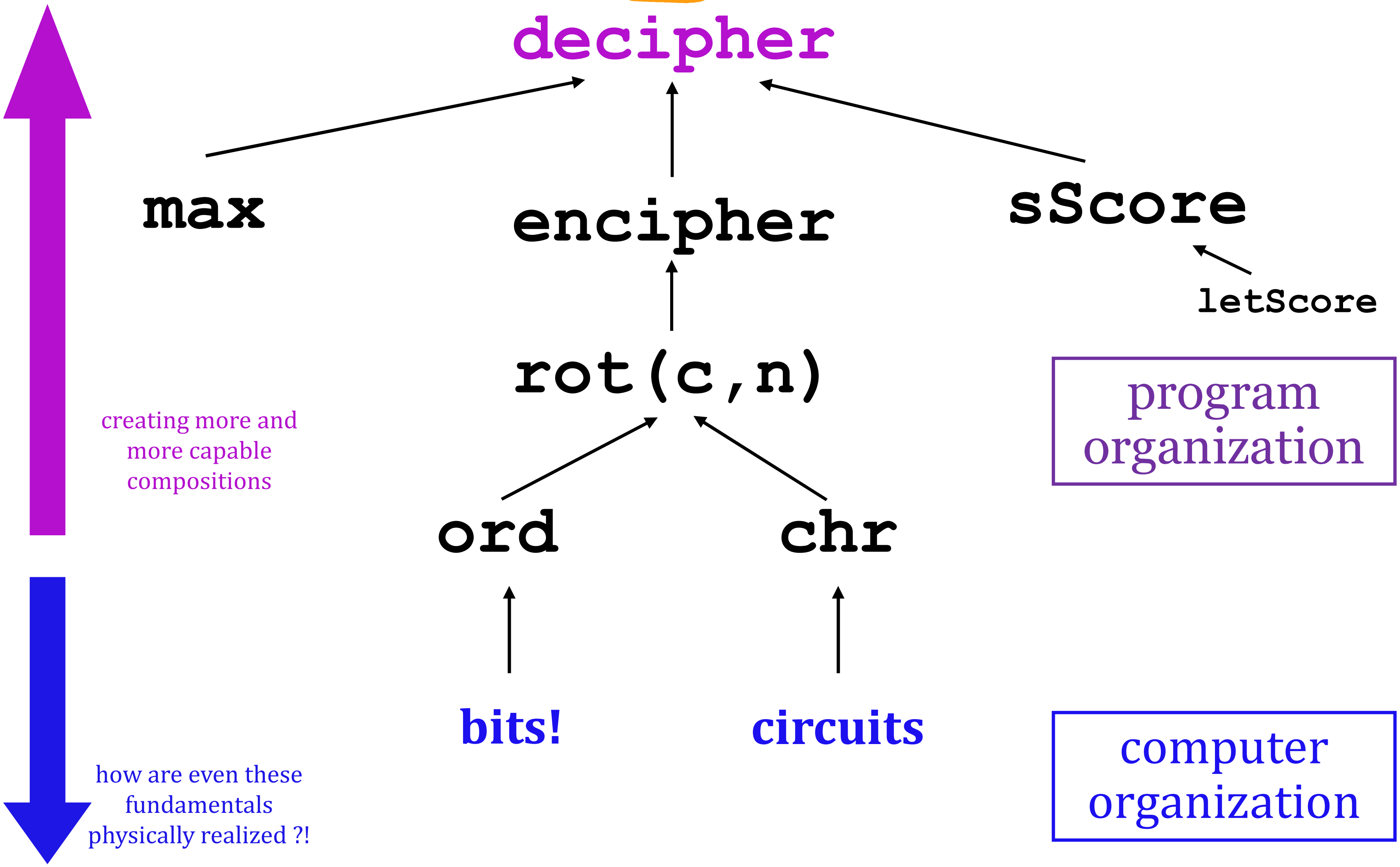
chr

creating more and
more capable
compositions

program
organization

Some legs to stand on!

It looks like I'm ahead of this...



decipher

max

encipher

sScore

letScore

rot(c,n)

ord

chr

bits!

circuits

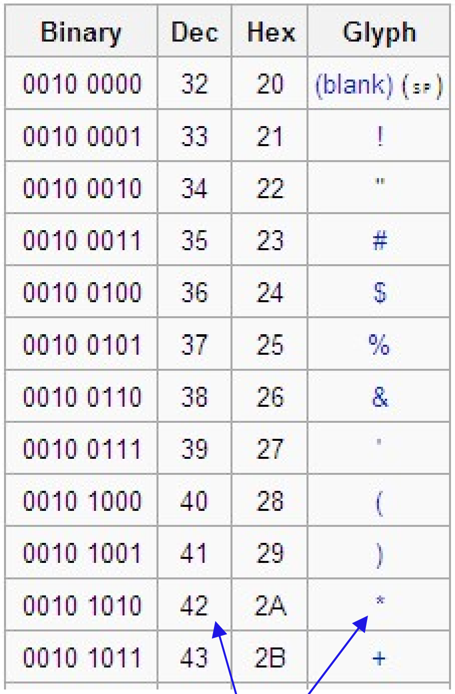
creating more and more capable compositions

how are even these fundamentals physically realized ?!

program organization

computer organization

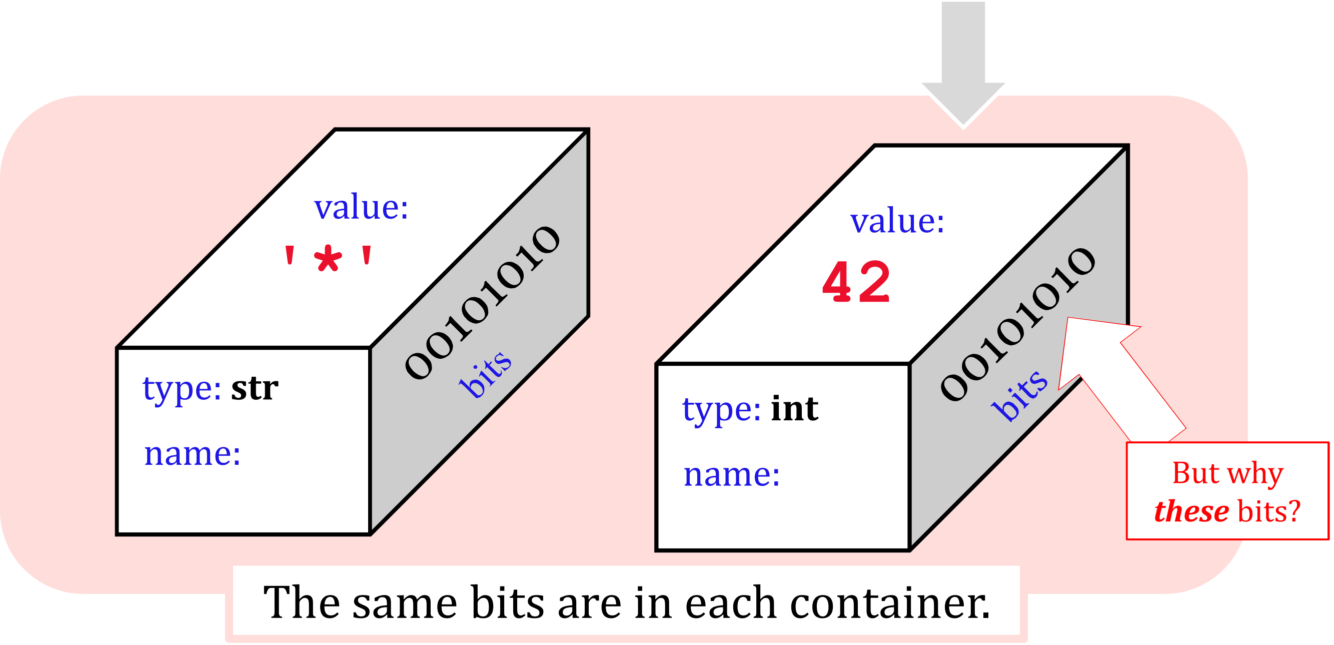
Binary Storage & Representation



Binary	Dec	Hex	Glyph
0010 0000	32	20	(blank) (␠)
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+

The SAME bits can represent different pieces of data, depending on **type**

8 bits = 1 byte = 1 box



value: '*'

00101010 bits

type: **str**

name:

value: 42

00101010 bits

type: **int**

name:

But why ***these*** bits?

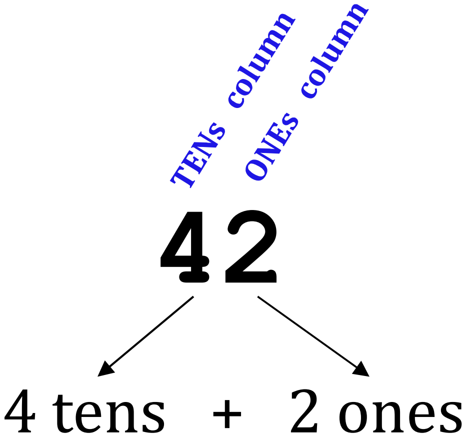
The same bits are in each container.

`42`

What *is* 42 ?

Base 10

What *is* 42 ?

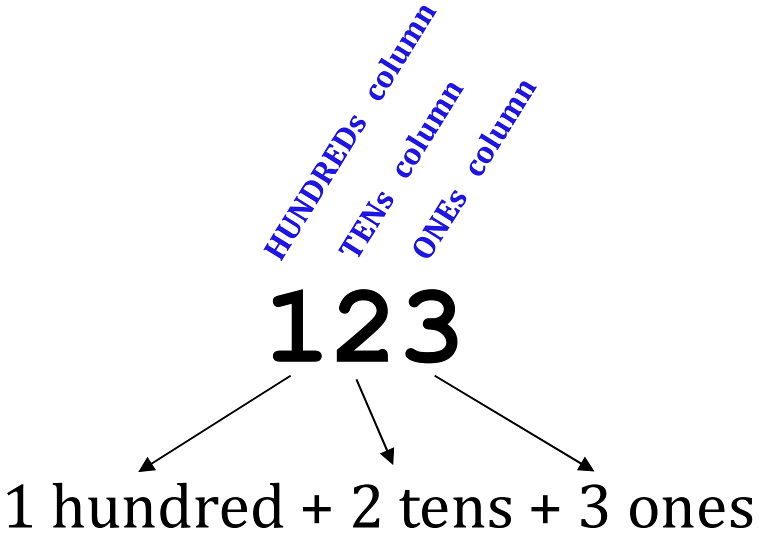


TENs column

ONEs column

42

4 tens + 2 ones



HUNDREDs column

TENs column

ONEs column

123

1 hundred + 2 tens + 3 ones

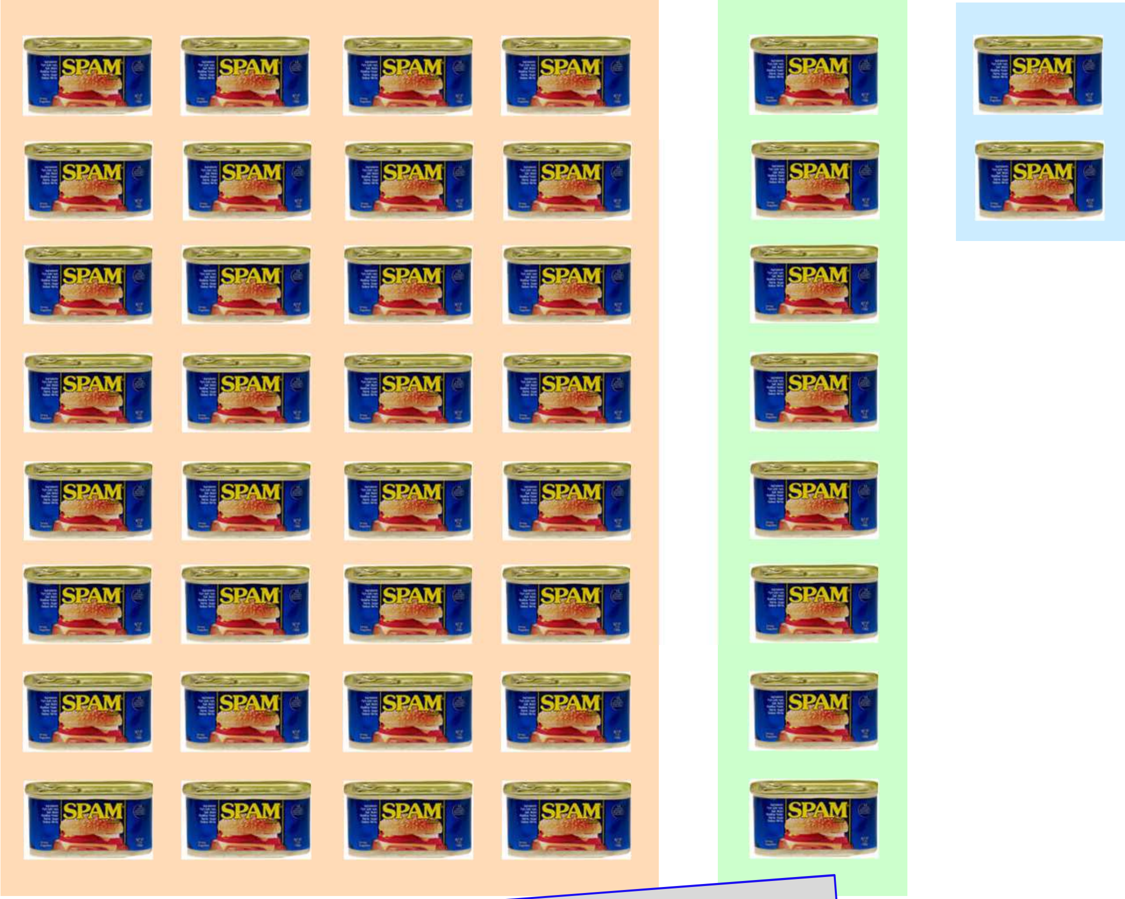
42

Value (semantics)

stuff we care about (what things *mean)*

Syntax

stuff we need *to communicate*



SAME!

101010

different

Value (semantics)

stuff we care about (what things *mean)*

Syntax

stuff we need *to communicate*

Base 2

THIRTYTWOs col. SIXTEENs col. EIGHTs column FOURs column TWOs column ONEs column

101010_2

128's column SIXTYFOURs col THIRTYTWOs col. SIXTEENs col. EIGHTs column FOURs column TWOs column ONEs column

_ _ _ _ _ _ _ _

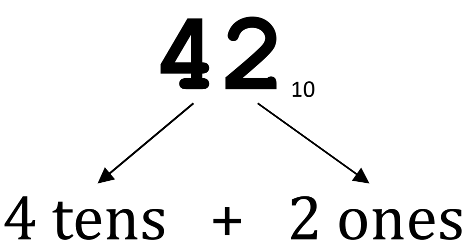
Write 123 in binary...

each column represents the base's next power

123

Base 10

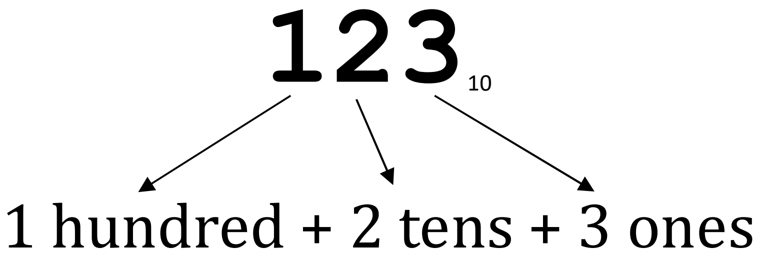
TENs column ONEs column



42_{10}

4 tens + 2 ones

HUNDREDs column TENs column ONEs column



123_{10}

1 hundred + 2 tens + 3 ones

Base 2

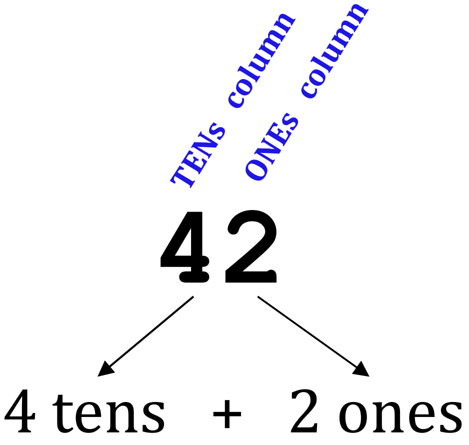
Base 10

each column represents the base's next power

THIRTYTWOs col.
SIXTEENs col.
EIGHTs column
FOURs column
TWOs column
ONEs column

101010

$32 + 8 + 2 = 42_{10}$

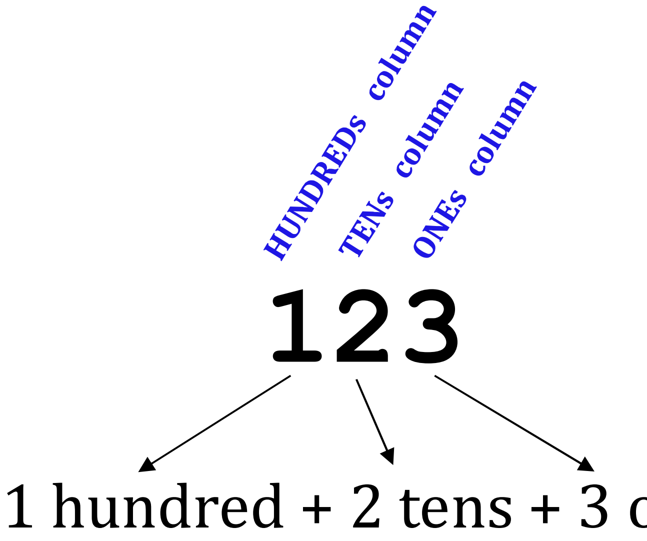


128's column
SIXTYFOURs col
THIRTYTWOs col.
SIXTEENs col.
EIGHTs column
FOURs column
TWOs column
ONEs column

$01111011_2 = 123_{10}$

_ _ _ _ _ _ _ _

Write 123 in binary...



1 1
123
-64
59
-32
27
-16
11
-8
3

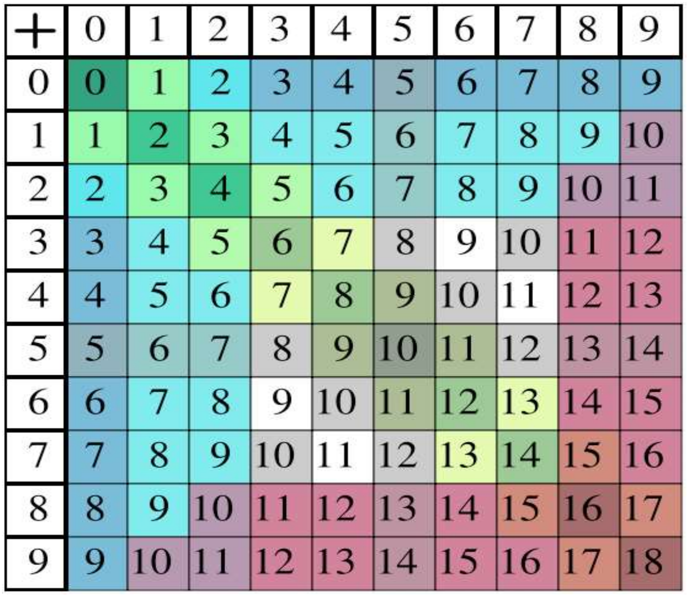
Binary math

Decimal math

tables of basic facts

+

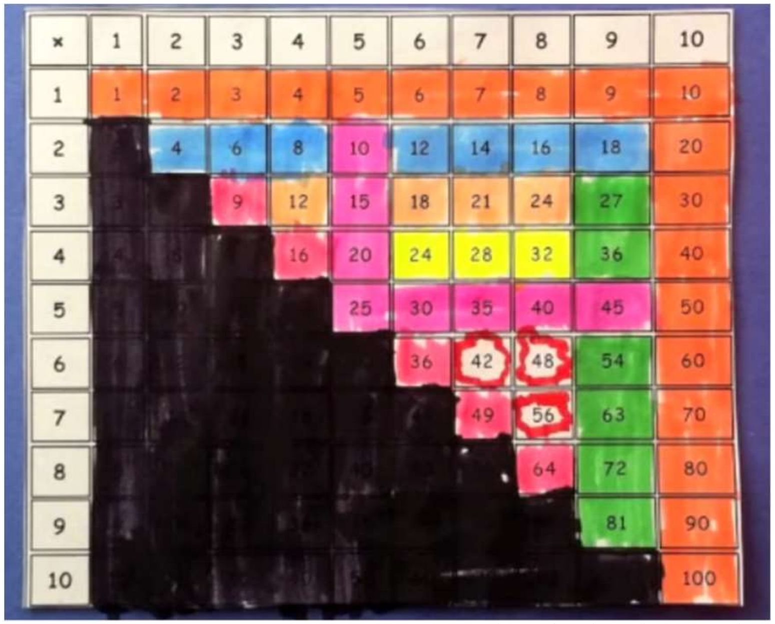
Addition



+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

*

Multiplication



×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	[redacted]	4	6	8	10	12	14	16	18	20
3	[redacted]	[redacted]	9	12	15	18	21	24	27	30
4	[redacted]	[redacted]	[redacted]	16	20	24	28	32	36	40
5	[redacted]	[redacted]	[redacted]	[redacted]	25	30	35	40	45	50
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	36	42	48	54	60
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	49	56	63	70
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	64	72	80
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	81	90
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	100

www.youtube.com/watch?v=Nh7xapVB-Wk

Binary math

Decimal math

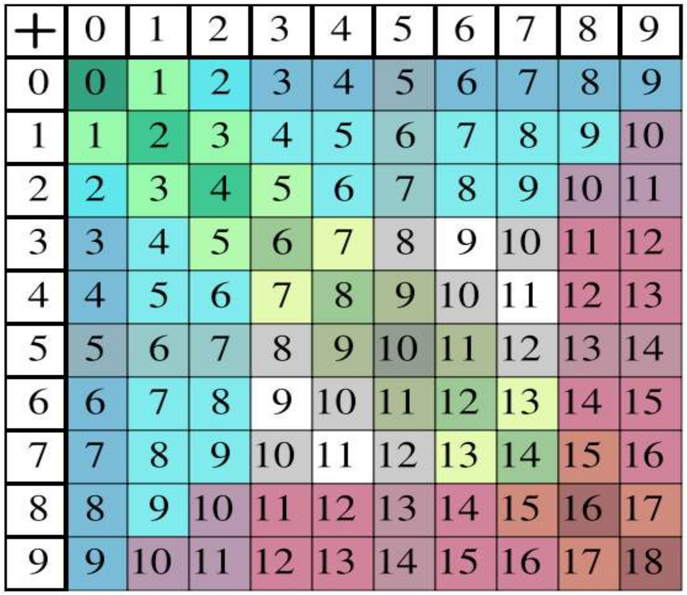
tables of basic facts

+	0	1
0	0	1
1	1	10

twos ones

+

Addition

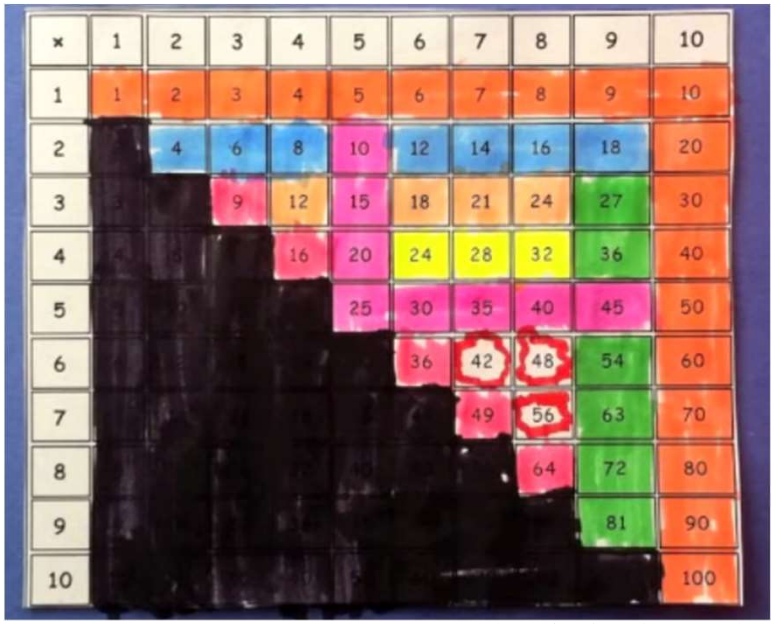


+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

*	0	1
0	0	0
1	0	1

*

Multiplication



×	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	[redacted]	4	6	8	10	12	14	16	18	20
3	[redacted]	[redacted]	9	12	15	18	21	24	27	30
4	[redacted]	[redacted]	[redacted]	16	20	24	28	32	36	40
5	[redacted]	[redacted]	[redacted]	[redacted]	25	30	35	40	45	50
6	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	36	42	48	54	60
7	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	49	56	63	70
8	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	64	72	80
9	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	81	90
10	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	100

www.youtube.com/watch?v=Nh7xapVB-Wk

Name(s): ______________________________

Quiz

In binary, I'm an 11-eyed alien!

Convert these two binary numbers *to decimal*:

```
32 16  8  4  2  1
 1  1  0  0  1  1          10001000
```

Convert these two decimal numbers *to binary*:

```
32 16  8  4  2  1
```

28_{10} 101_{10}

Add these two binary numbers:

```
  101101
+   1110
--------
```

Multiply these binary numbers:

```
  101101
*   1110
--------



+
```

WITHOUT converting to decimal !

Hint: Remember these algorithms? They're the same in binary!

```
    1
  529
+ 742
-----
 1271
```

```
   529
*   42
------
  1058
+ 2116
------
 22218
```

Extra! Can you figure out the last binary digit (bit) of 53 ***without determining any other bits?*** The last two? 3?

Convert these two binary numbers ***to decimal***:

32	16	8	4	2	1
1	1	0	0	1	1

32 + 16 + 2 + 1

51

128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	0

128 + 8

136

values in blue

Convert these two decimal numbers ***to binary***:

28

32	16	8	4	2	1
0	1	1	1	0	0

101_{10}

128	64	32	16	8	4	2	1
0	1	1	0	0	1	0	1

syntax in orange

Extra! Can you figure out the last binary digit (bit) of 53 ***without determining any other bits***? The last *two*? *3*?

We'll return to this ***in a bit***...

Add these two binary numbers
WITHOUT converting to decimal !

```
          32  16  8  4  2  1

              1 0 1 1 0 1       45
+                 1 1 1 0       14
_____________________________

                                59

          32  16  8  4  2  1
```

Hint:

```
    1
   529
+  742
 -----
  1271
```

Do you remember this algorithm? It's the same!

Add these two binary numbers
WITHOUT converting to decimal !

$$\begin{array}{cccccccc}
 & & 1 & 1 & & & & \\
 & 32 & 16 & 8 & 4 & 2 & 1 & \\
 & 1 & 0 & 1 & 1 & 0 & 1 & 45 \\
+ & & & 1 & 1 & 1 & 0 & 14 \\ \hline
 & 1 & 1 & 1 & 0 & 1 & 1 & = 59_{10}
\end{array}$$

Hint:

$$\begin{array}{r}
\scriptstyle 1 \\
529 \\
+\ 742 \\ \hline
1271
\end{array}$$

Do you remember this algorithm? It's the same!

10_2 2_{10}

Multiply these two binary numbers
WITHOUT converting to decimal !

```
      32 16  8  4  2  1
       1  0  1  1  0  1      45
*           1  1  1  0      14
-----------------------
```

Hint:

```
   529
*   42
------
  1058
+2116
------
 22218
```

Do you remember this algorithm? It's the same!

630

Goal

A machine could -
and probably ***should***
- **be doing this !**

Multiply these two binary numbers
WITHOUT converting to decimal !

```
          32 16  8  4  2  1
           1  0  1  1  0  1      45
*                1  1  1  0      14
-------------------------------
           0  0  0  0  0  0
        1  0  1  1  0  1  0
     1  0  1  1  0  1  0  0
+ 1  0  1  1  0  1  0  0  0
-------------------------------
1  0  0  1  1  1  0  1  1  0     630
512 256 128 64 32 16 8  4  2  1
```

"partial products"

Goal

Hint:

```
    529
*    42
-------
   1058
+ 2116
-------
  22218
```

Do you remember this algorithm? It's the same!

A machine could -
*and probably **should***
- **be doing this !**

Beyond Binary

base 1 111111111111111111111111111111111111111111 digits: **1**

```
                32  16  8   4   2   1
base 2 ——————  1   0   1   0   1   0
```

digits: **0, 1**

```
                       27  9   3   1
base 3 ——————————
```

digits: **0, 1, 2**

42 ?

There are 10 kinds of "people" in the universe:
those who know ternary,
those who don't, and
those who think this is a binary joke!

Beyond Binary

base 1 111111111111111111111111111111111111111111 digits: 1

32 16 8 4 2 1

base 2 101010 digits: 0, 1

27 9 3 1

base 3 1120 digits: 0, 1, 2

16 4 1

base 4

base 5

base 6

base 7

base 8

base 9

base 10 42 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

base 11

base 12

. . .

base 16

Which of these ***isn't*** 42...?

222 60 54 46 39

and what are the ***bases*** of the rest?

Beyond Binary

base 1 111111111111111111111111111111111111111111 digits: 1

base 2 — 32 16 8 4 2 1

101010 digits: 0, 1

base 3 — 27 9 3 1

1120 digits: 0, 1, 2

base 4 16 4 1

base 5

base 6

base 7

base 8

base 9

base 10 42 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

base 11

base 12

. . .

base 16

16s 4s 1s 7s 1s 9s 1s 11s 1s

Which of these *isn't* 42…?

222 60_7 54 46_9 39_{11}

and what are the *bases* of the rest?

1s 4s 16 42

base 1		111111111111111111111111111111111111111111	digits: 1
base 2	128 64 32 16 8 4 2 1	101010	digits: 0, 1
base 3	81 27 9 3 1	1120	digits: 0, 1, 2
base 4	64 16 4 1	222	digits: 0, 1, 2, 3
base 5	125 25 5 1	132	digits: 0, 1, 2, 3, 4
base 6	216 36 6 1	110	digits: 0, 1, 2, 3, 4, 5
base 7	49 7 1	60	digits: 0, 1, 2, 3, 4, 5, 6
base 8	64 8 1	52	digits: 0, 1, 2, 3, 4, 5, 6, 7
base 9	81 9 1	46	digits: 0, 1, 2, 3, 4, 5, 6, 7, 8
base 10	100 10 1	42	digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
base 11	121 11 1	39	digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A
base 16	256 16 1	2A	Hexadecimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

All 42s!

base 1 111111111111111111111111111111111111111111 digits: 1

All 42s!

128 64 32 16 8 4 2 1

base 2 —— 101010 digits: 0, 1

81 27 9 3 1

base 3 —— 1120 digits: 0, 1, 2

Hexadecimal

256 16 1

base 16 —— 2A digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

base 1 111111111111111111111111111111111111111111

digits: **1**

128 64 32 16 8 4 2 1

base 2 —— **101010**

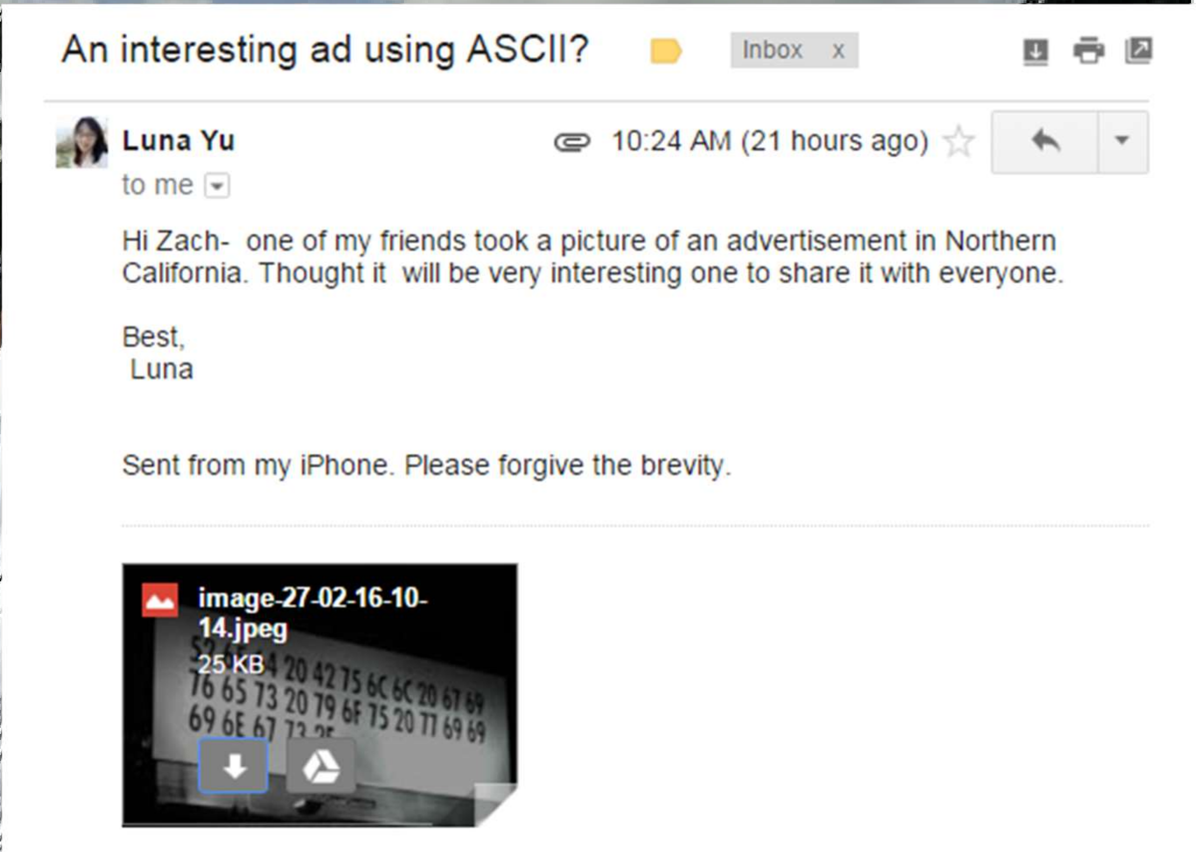
digits: **0, 1**

81 27 9 3 1

base 3 —— **1120**

digits: **0, 1, 2**

All 42s!



An interesting ad using ASCII?

Luna Yu 10:24 AM (21 hours ago)

to me

Hi Zach- one of my friends took a picture of an advertisement in Northern California. Thought it will be very interesting one to share it with everyone.

Best,
Luna

Sent from my iPhone. Please forgive the brevity.

image-27-02-16-10-14.jpeg
25 KB

256 16 1

base 16 —— **2A**

Hexadecimal

digits: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

base 1 111111111111111111111111111111111111111111

digits: **1**

base 2 ——— 101010

128 64 32 16 8 4 2 1

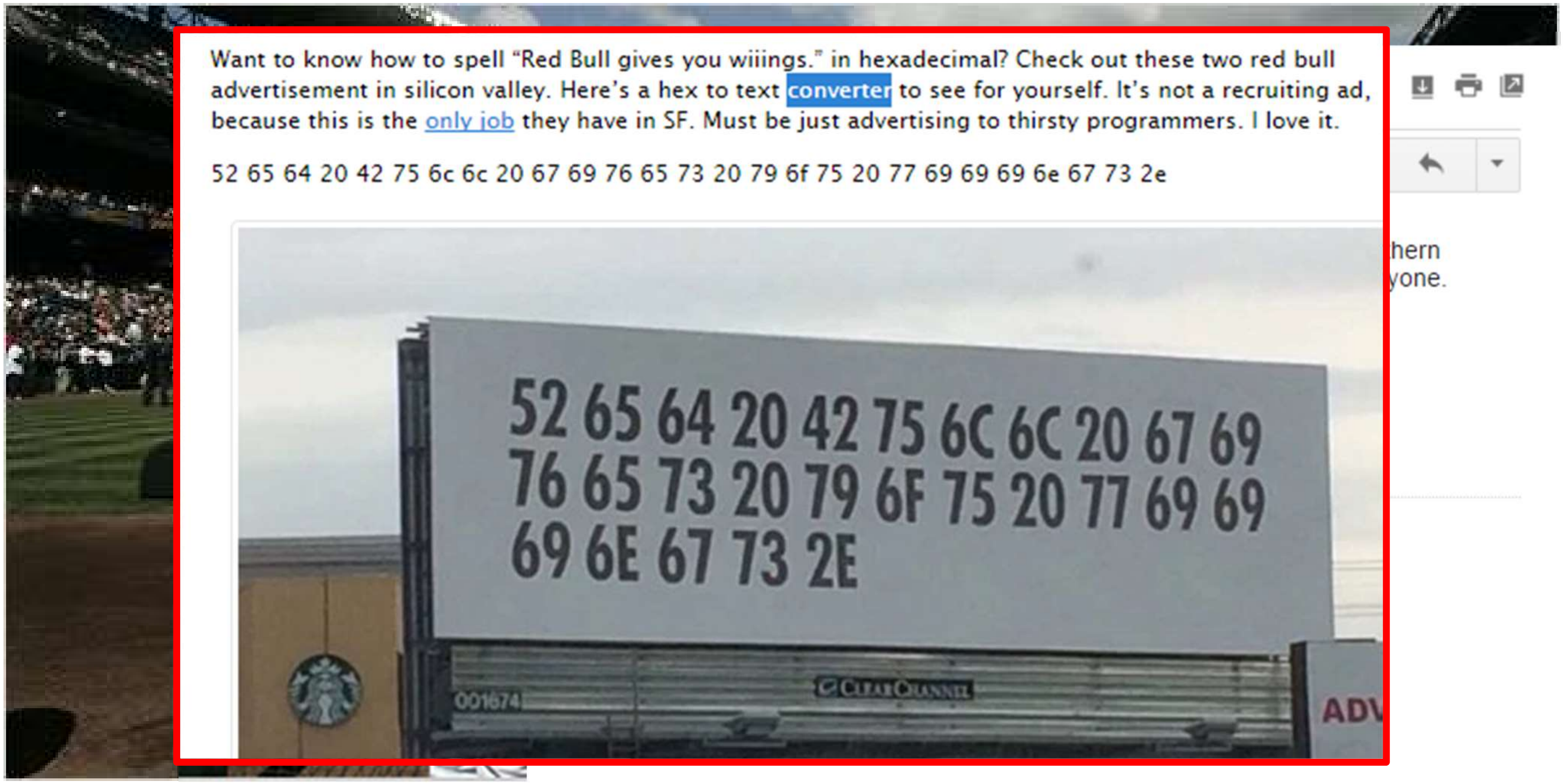
digits: **0, 1**

base 3 ——— 1120

81 27 9 3 1

digits: **0, 1, 2**

All 42s!



Want to know how to spell “Red Bull gives you wiiings.” in hexadecimal? Check out these two red bull advertisement in silicon valley. Here’s a hex to text converter to see for yourself. It’s not a recruiting ad, because this is the only job they have in SF. Must be just advertising to thirsty programmers. I love it.

52 65 64 20 42 75 6c 6c 20 67 69 76 65 73 20 79 6f 75 20 77 69 69 69 6e 67 73 2e

52 65 64 20 42 75 6C 6C 20 67 69
76 65 73 20 79 6F 75 20 77 69 69
69 6E 67 73 2E
CLEAR CHANNEL
001674
ADV
hern
yone.

base 16 ——— 2A

256 16 1

Hexadecimal

digits: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

base 1 111111111111111111111111111111111111111111

digits: **1**

128 64 32 16 8 4 2 1

base 2 —— **101010**

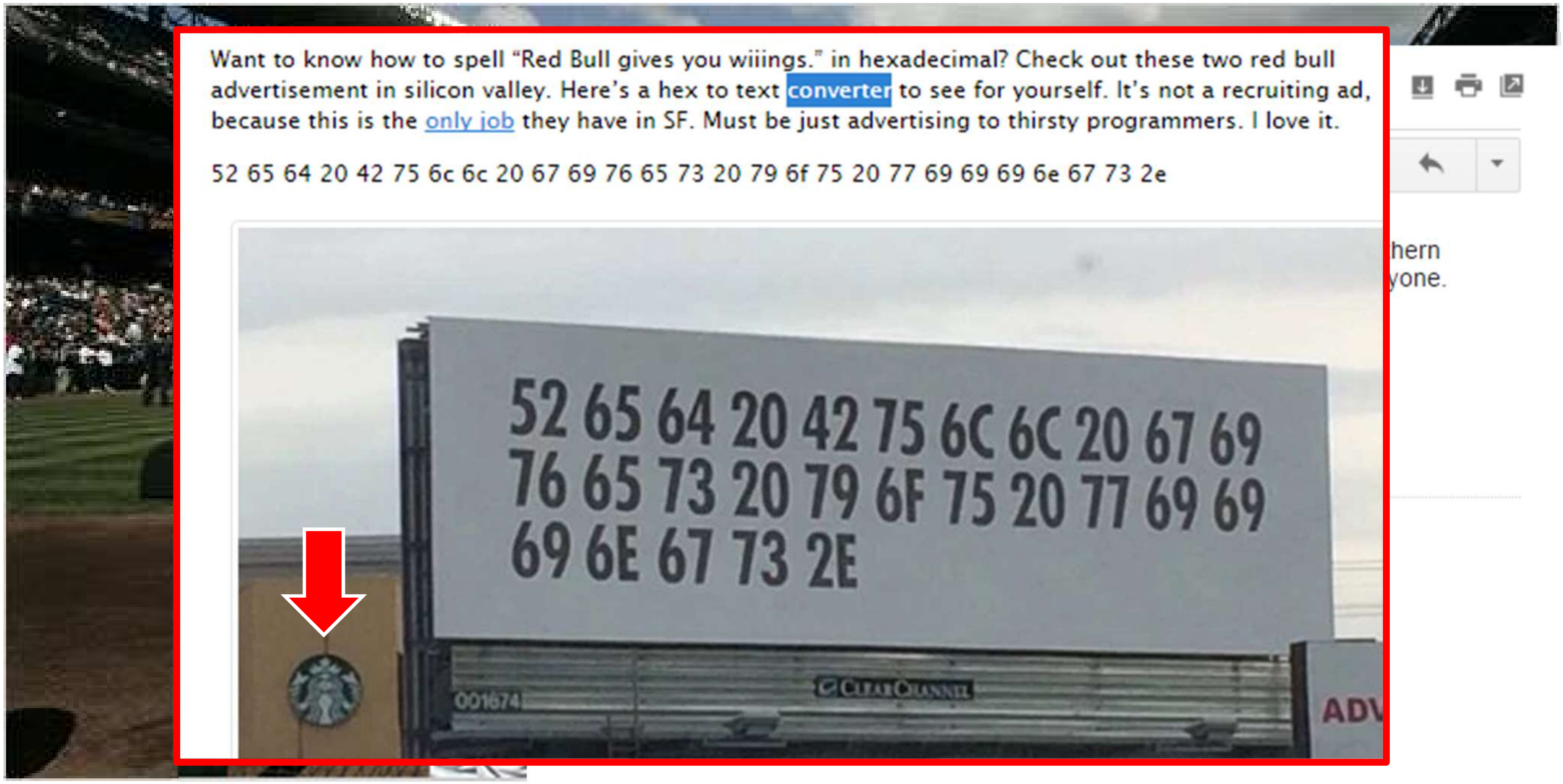
digits: **0, 1**

81 27 9 3 1

base 3 —— **1120**

digits: **0, 1, 2**

All 42s!



Want to know how to spell “Red Bull gives you wiiings.” in hexadecimal? Check out these two red bull advertisement in silicon valley. Here’s a hex to text converter to see for yourself. It’s not a recruiting ad, because this is the only job they have in SF. Must be just advertising to thirsty programmers. I love it.

52 65 64 20 42 75 6c 6c 20 67 69 76 65 73 20 79 6f 75 20 77 69 69 69 6e 67 73 2e

52 65 64 20 42 75 6C 6C 20 67 69
76 65 73 20 79 6F 75 20 77 69 69
69 6E 67 73 2E
001674
CLEAR CHANNEL
ADV
hern
yone.

Hexadecimal

256 16 1

base 16 —— **2A**

digits: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**

Off base?

Our Mascot, the Panda

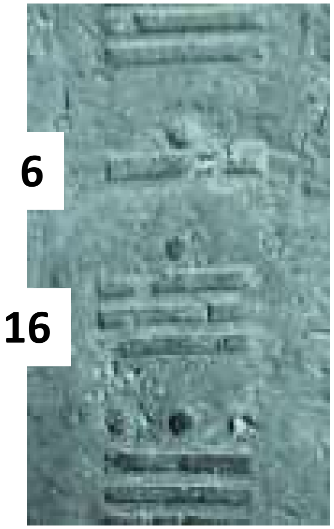
DOZENAL SOCIETY OF AMERICA
0
1
2
3
4
5
6
7
8
9
χ
Ɛ

Base 12 –

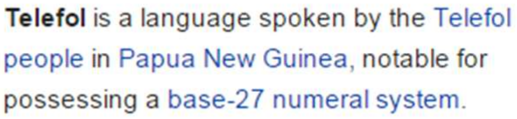
"~~Duodecimal~~ Society"

"***Dozenal*** Society"

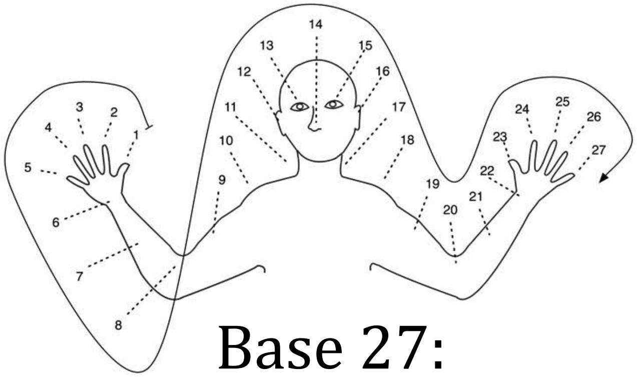
Base 20: Americas



Olmec base-20 numbers
E. Mexico, ~ 300 AD

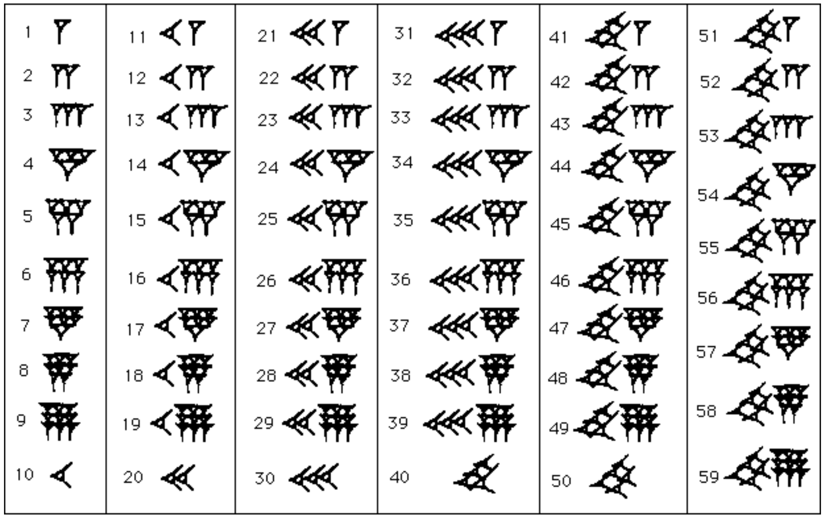


Telefol is a language spoken by the Telefol people in Papua New Guinea, notable for possessing a base-27 numeral system.



Base 27: New Guinea

Base 60 – Ancient Sumeria

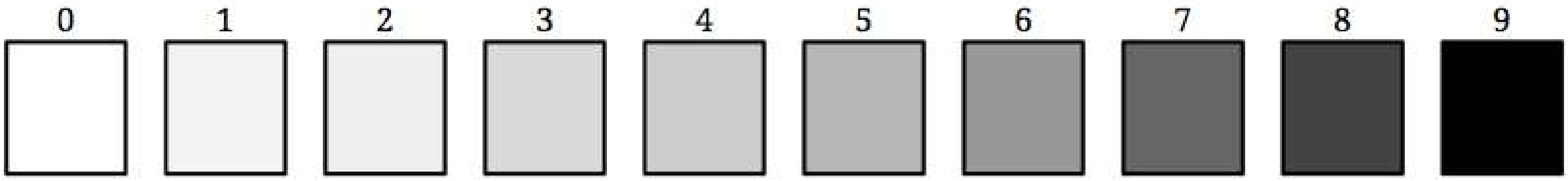

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

Some of these bases are still echoing around...

But ***why*** binary?

Ten symbols is too many!

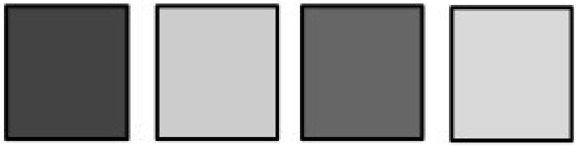
A computer has to differentiate *physically* among all its possibilities.



ten symbols ~ ten different voltages

This is too difficult to replicate billions of times

engineering!

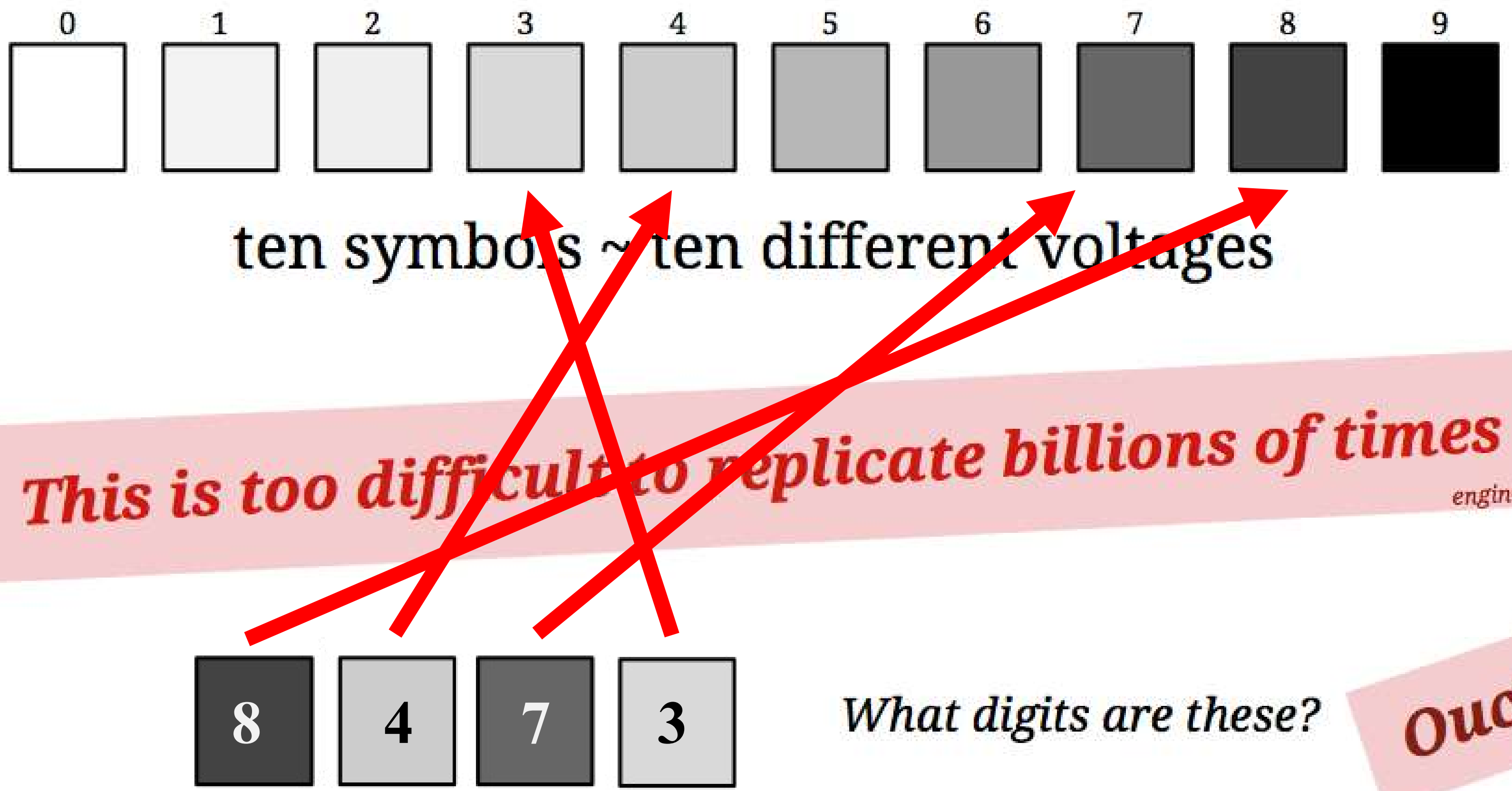


What digits are these?

Ouch!

Ten symbols is too many!

A computer has to differentiate *physically* among all its possibilities.



ten symbols ~ ten different voltages

This is too difficult to replicate billions of times

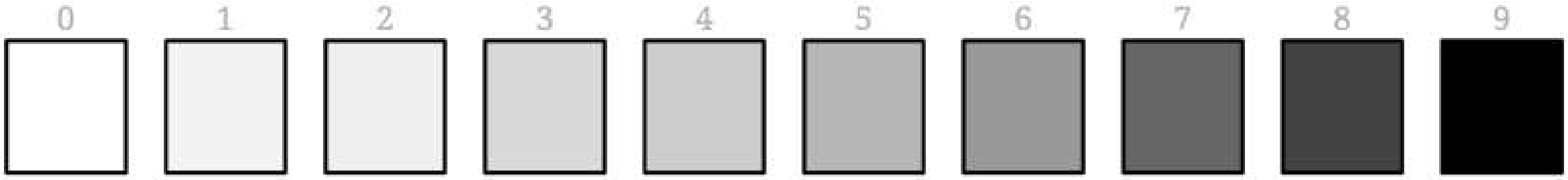
engineering!

What digits are these?

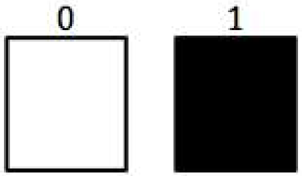
Ouch!

Two symbols is easiest!

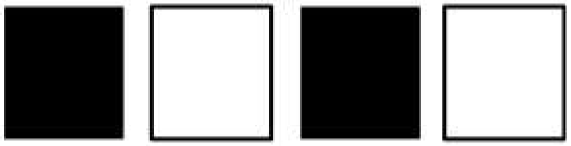
A computer has to differentiate *physically* among all its possibilities.



ten symbols ~ ten different voltages



two symbols ~ two different voltages

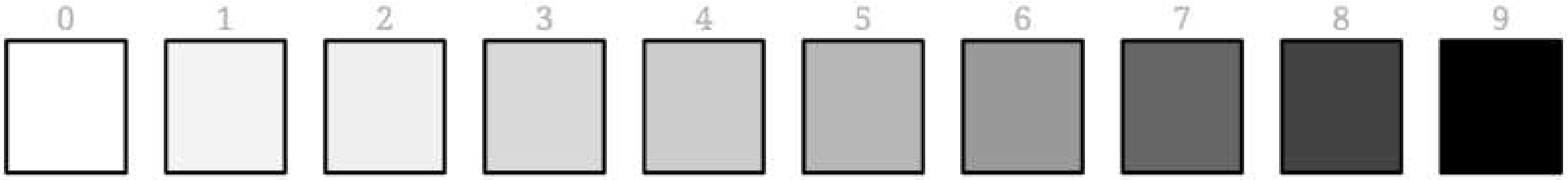


What digits are these?

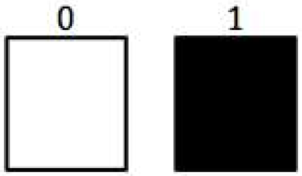
Easy!

Two symbols is easiest!

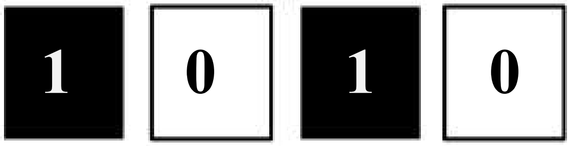
A computer has to differentiate *physically* among all its possibilities.



ten symbols ~ ten different voltages



two symbols ~ two different voltages

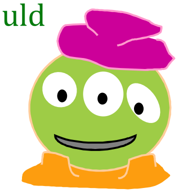


What digits are these?

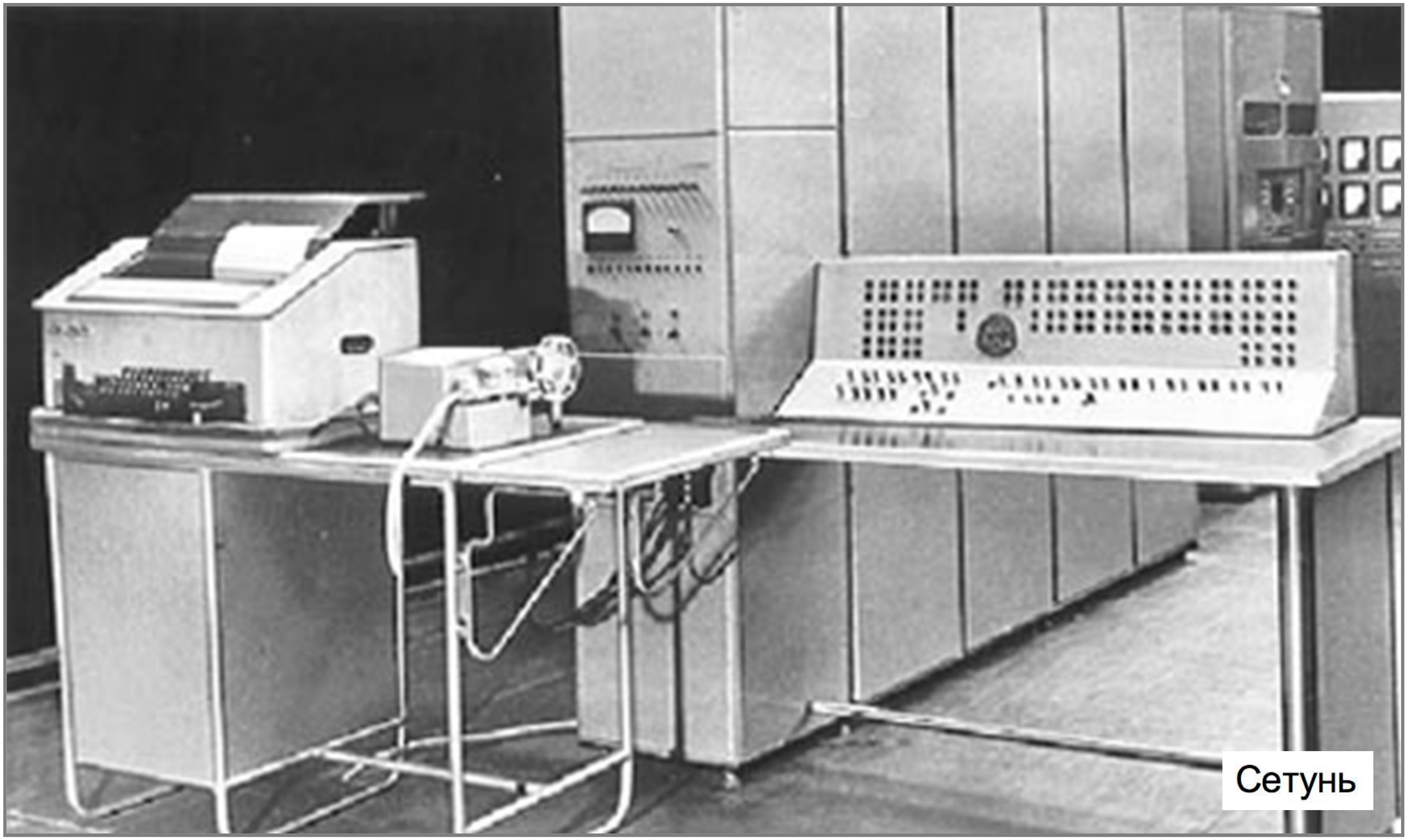
Easy!

Ternary computers?

Everything should be base-3!

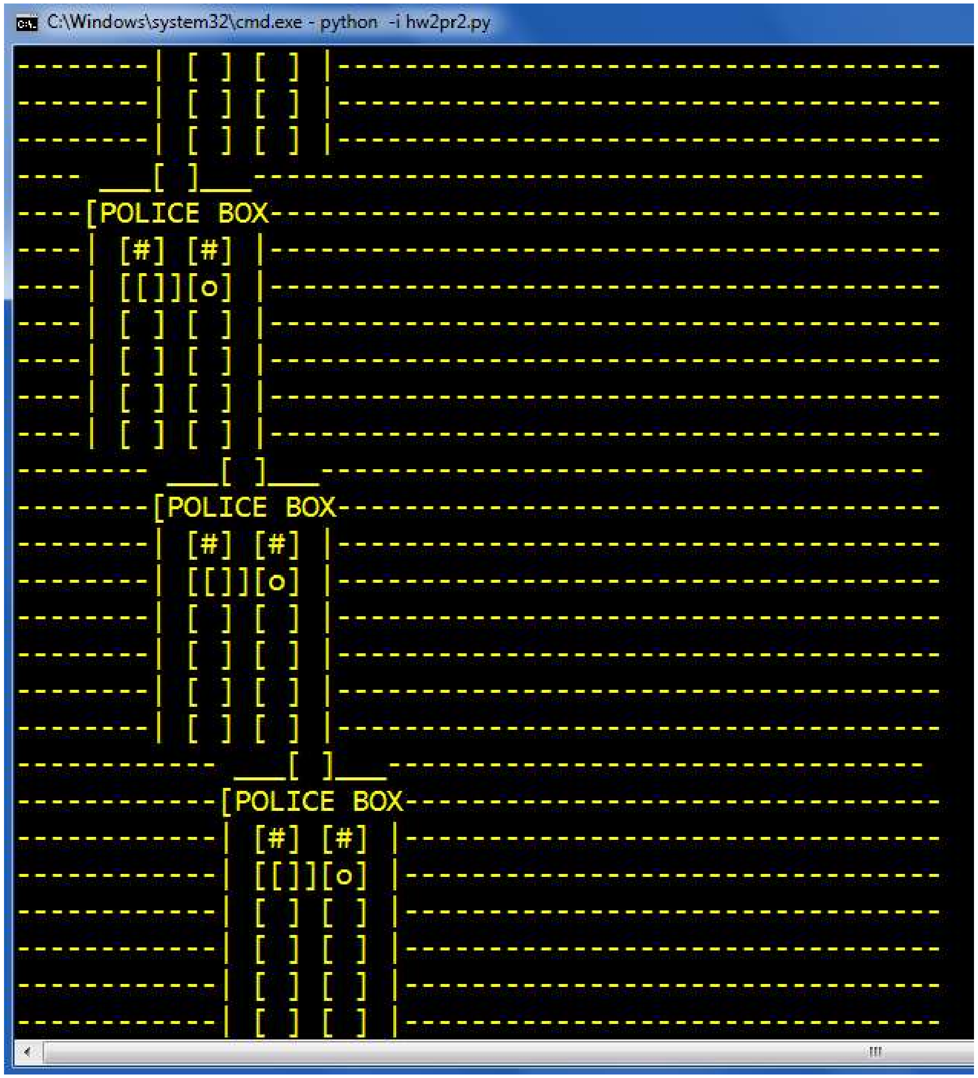


50 of these ***Setun*** ternary machines were made at Moscow U. ~ **1958**



This project was discontinued in 1970… ***though not because of the ternary design!***

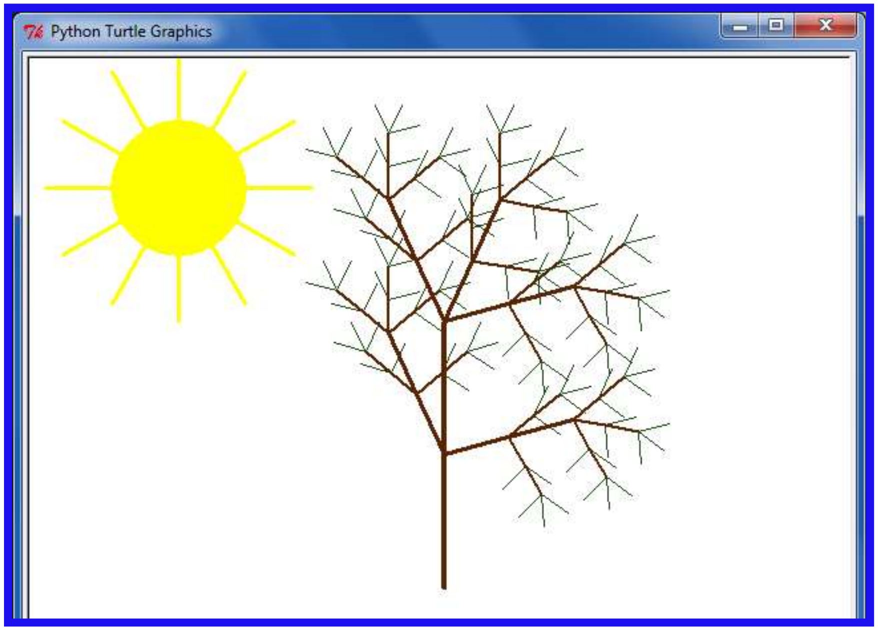
Eye-catching submissions...



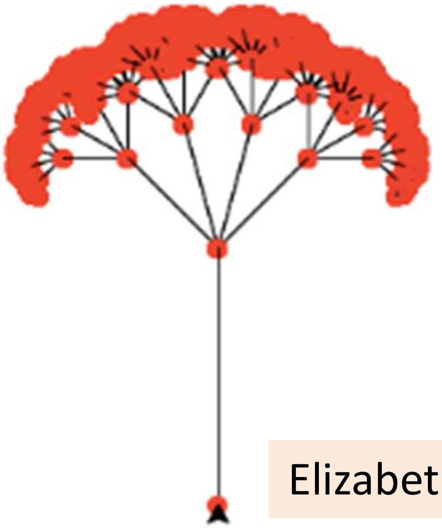
```
--------| [ ] [ ] |-------------------------------------
--------| [ ] [ ] |-------------------------------------
--------| [ ] [ ] |-------------------------------------
---- ___[ ]___-----------------------------------------
----[POLICE BOX-----------------------------------------
----| [#] [#] |-----------------------------------------
----| [[]][o] |-----------------------------------------
----| [ ] [ ] |-----------------------------------------
----| [ ] [ ] |-----------------------------------------
----| [ ] [ ] |-----------------------------------------
----| [ ] [ ] |-----------------------------------------
-------- ___[ ]___-------------------------------------
--------[POLICE BOX-------------------------------------
--------| [#] [#] |-------------------------------------
--------| [[]][o] |-------------------------------------
--------| [ ] [ ] |-------------------------------------
--------| [ ] [ ] |-------------------------------------
--------| [ ] [ ] |-------------------------------------
--------| [ ] [ ] |-------------------------------------
------------ ___[ ]___---------------------------------
------------[POLICE BOX---------------------------------
------------| [#] [#] |---------------------------------
------------| [[]][o] |---------------------------------
------------| [ ] [ ] |---------------------------------
------------| [ ] [ ] |---------------------------------
------------| [ ] [ ] |---------------------------------
------------| [ ] [ ] |---------------------------------
```

ASCII wanderings...

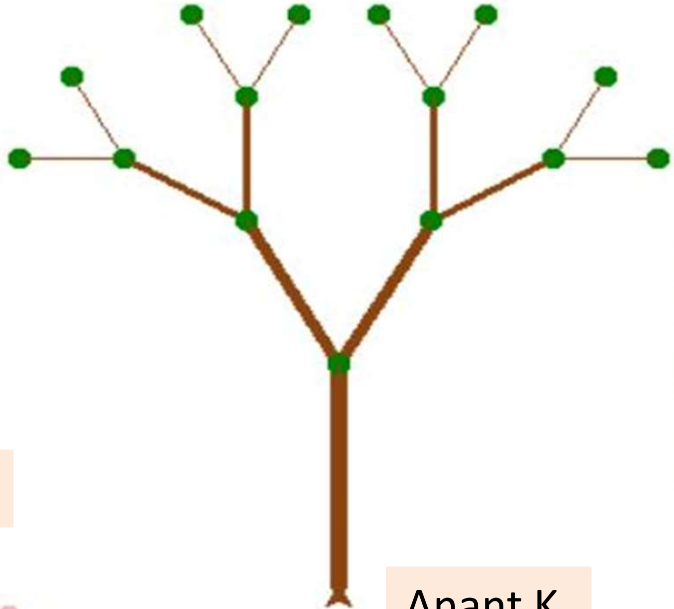
and turtle art



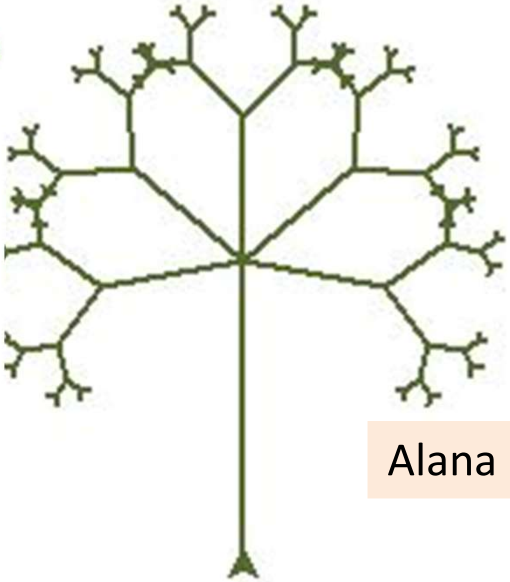
Trees…



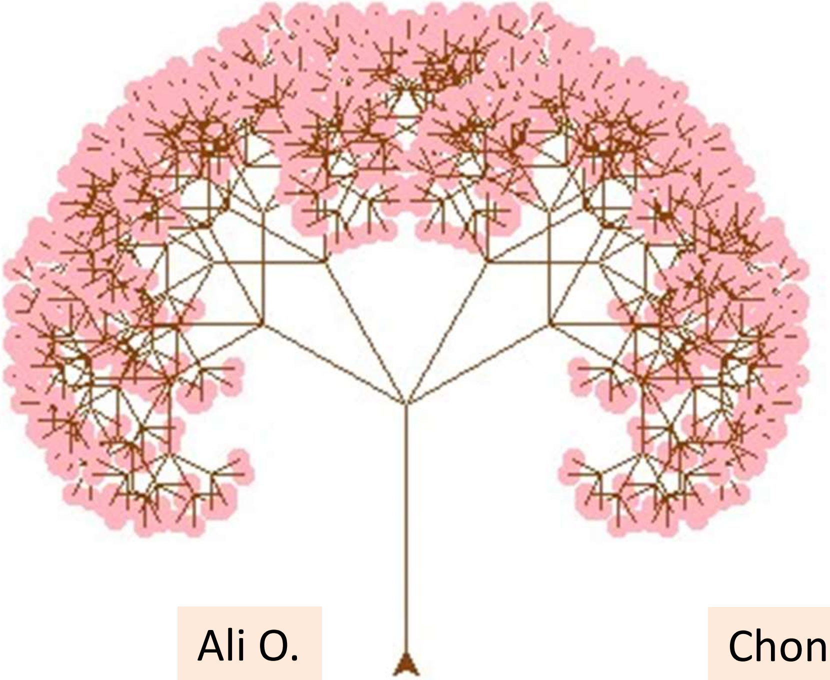
Elizabeth M.



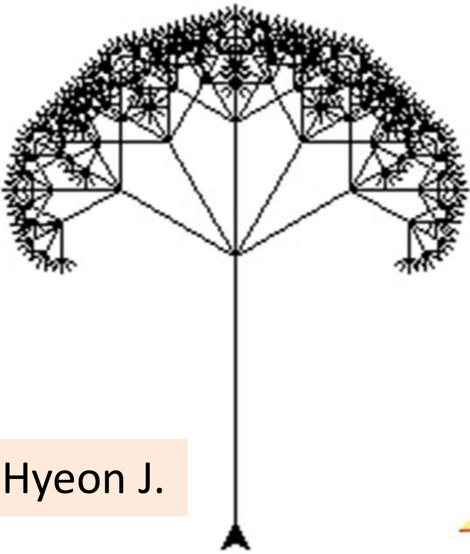
Anant K.



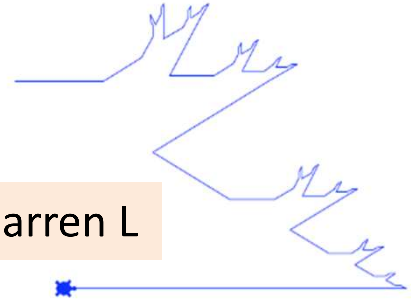
Alana G.



Ali O.

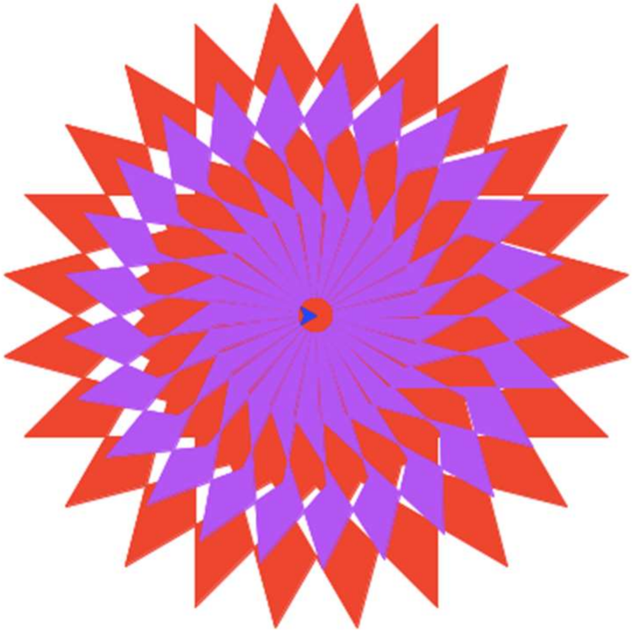


ChongHyeon J.

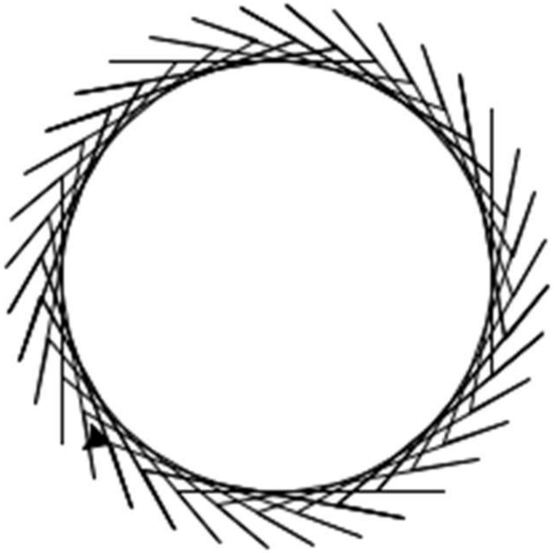


Darren L

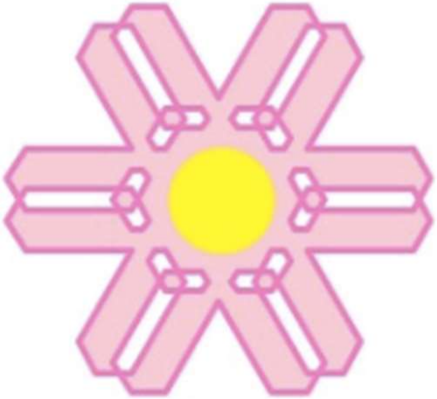
Yuexi L.



Yinfeng H.

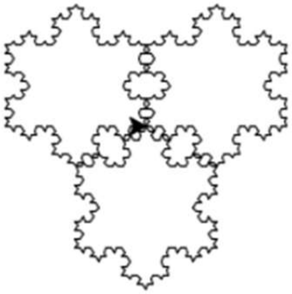


Arthur C.

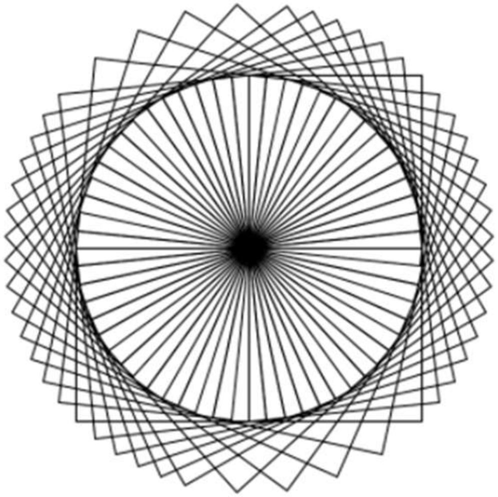


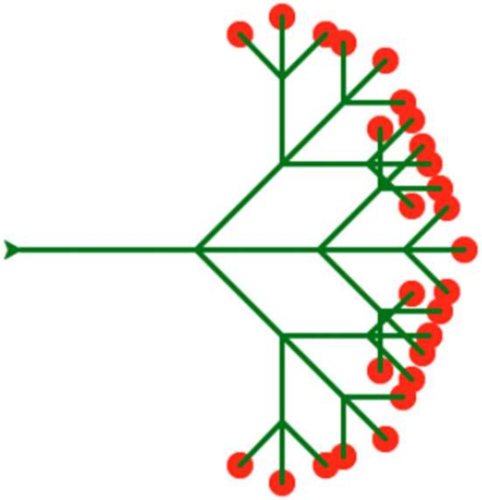
Roxy K.

Spirals...

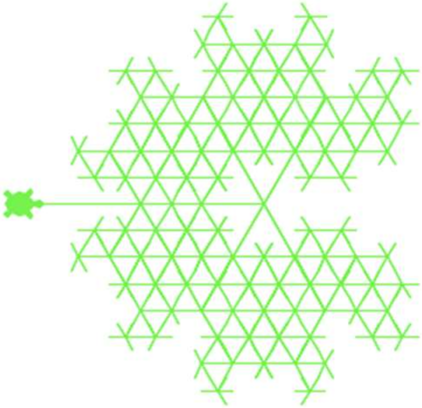


Indu S.

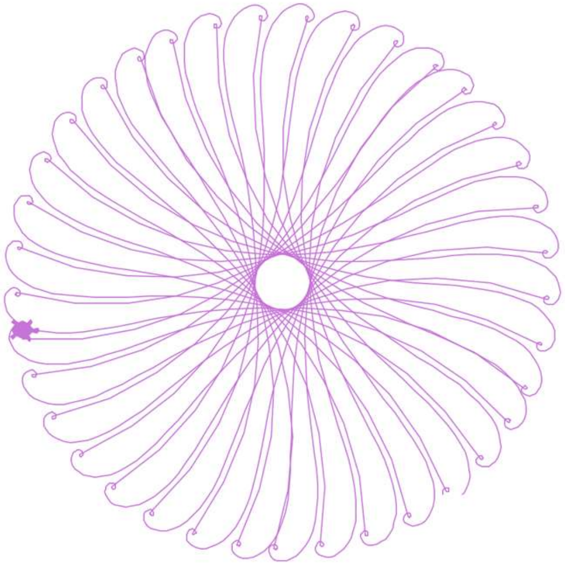




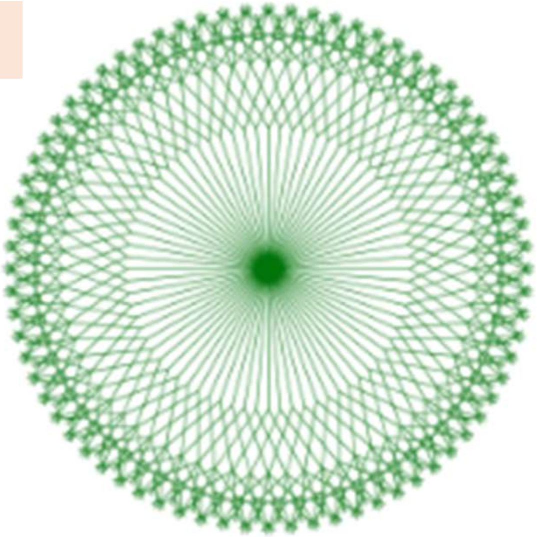
Shruthi T



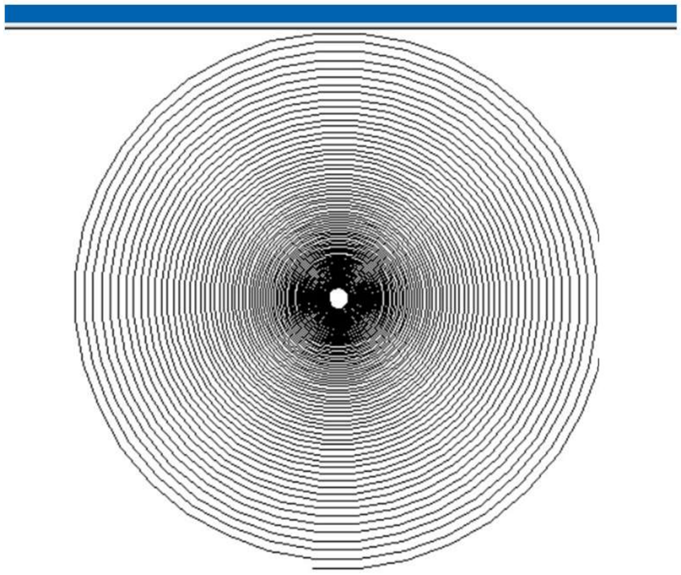
Shaneli J



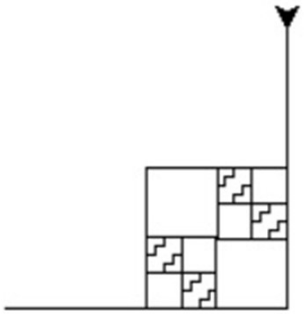
Madeline H

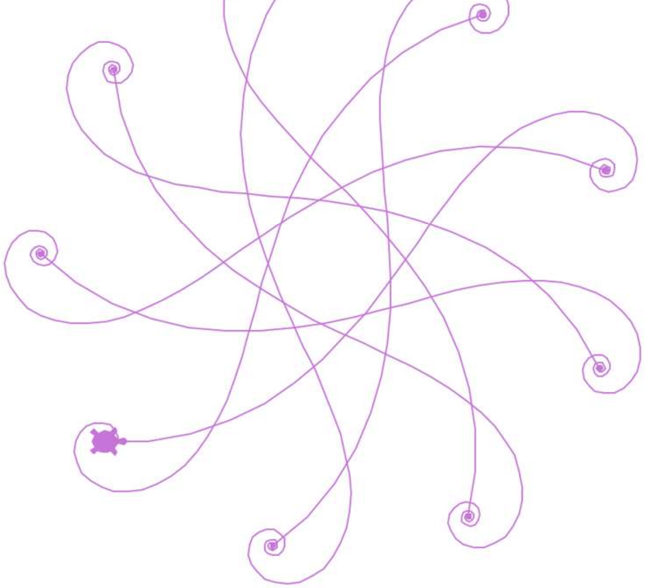


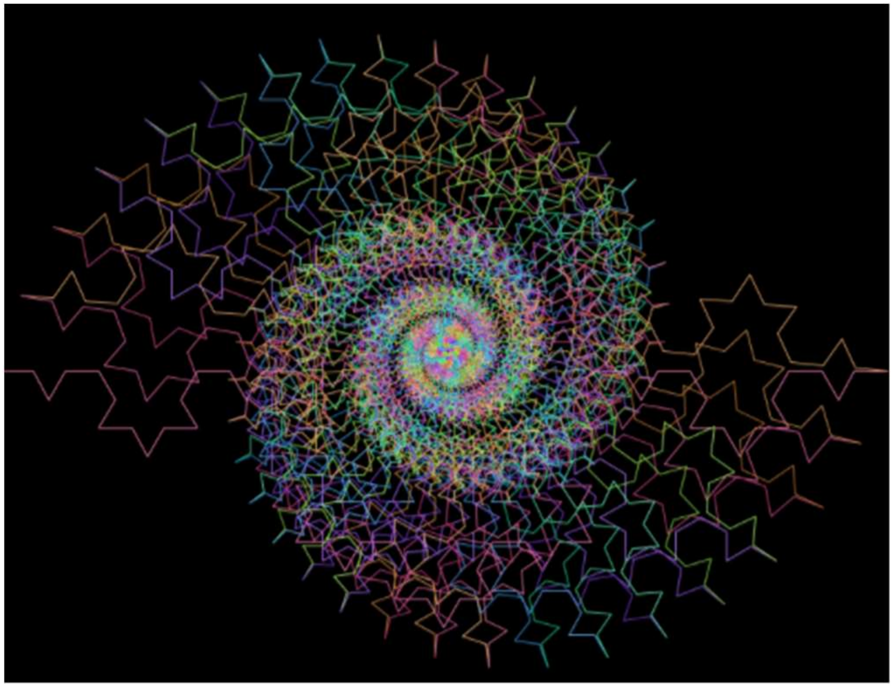
Daniel C.



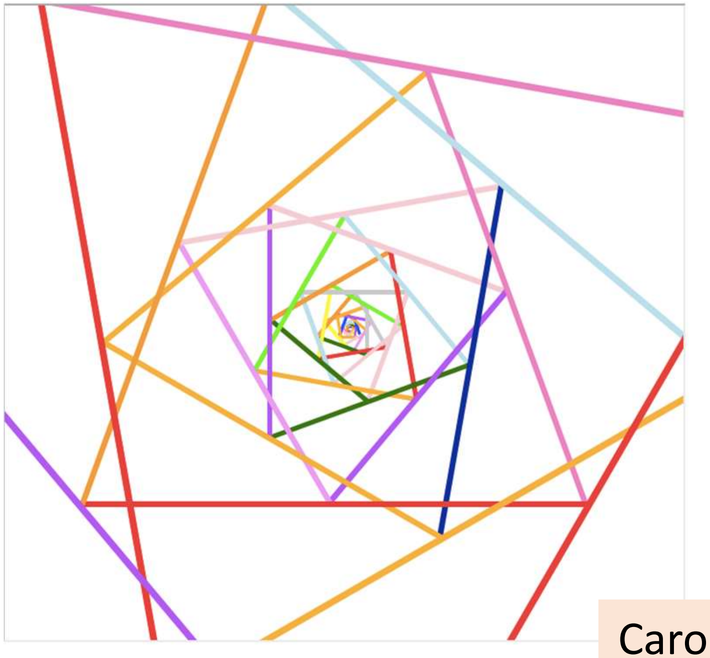
Jessica Y.





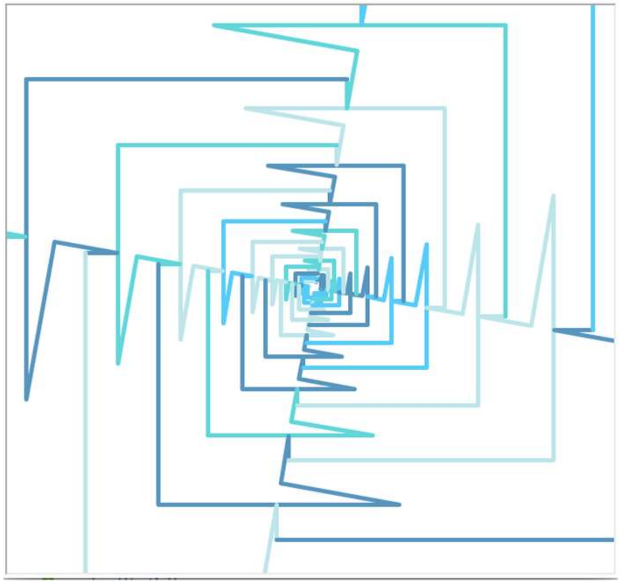


Seth T.B.

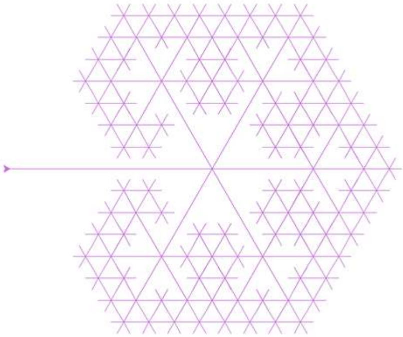


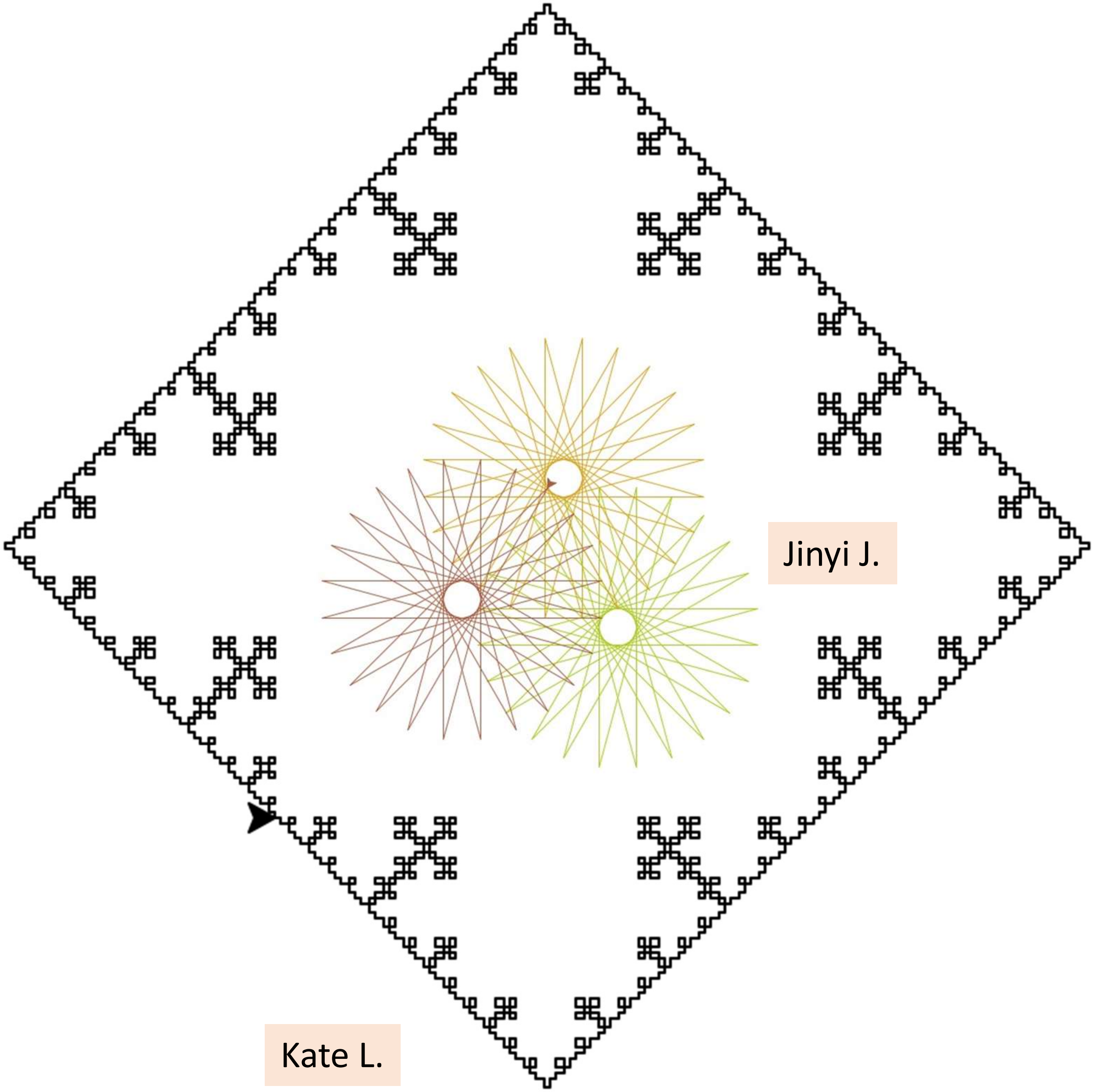
Caroline T

Emily Z



Both!?

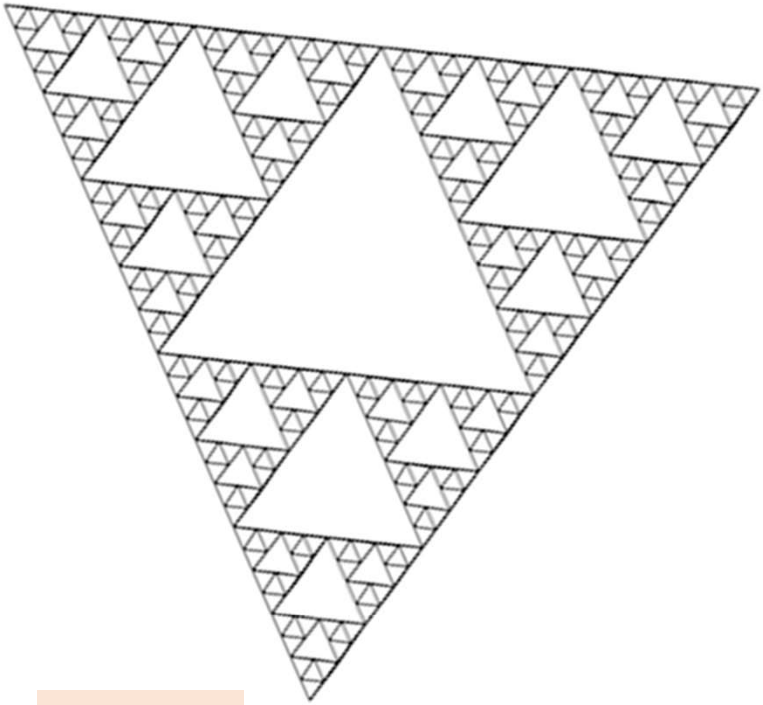




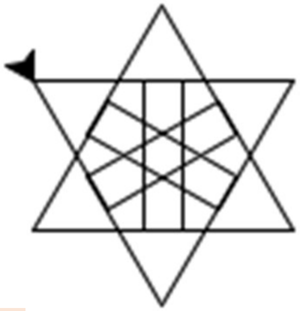
Jinyi J.

Kate L.

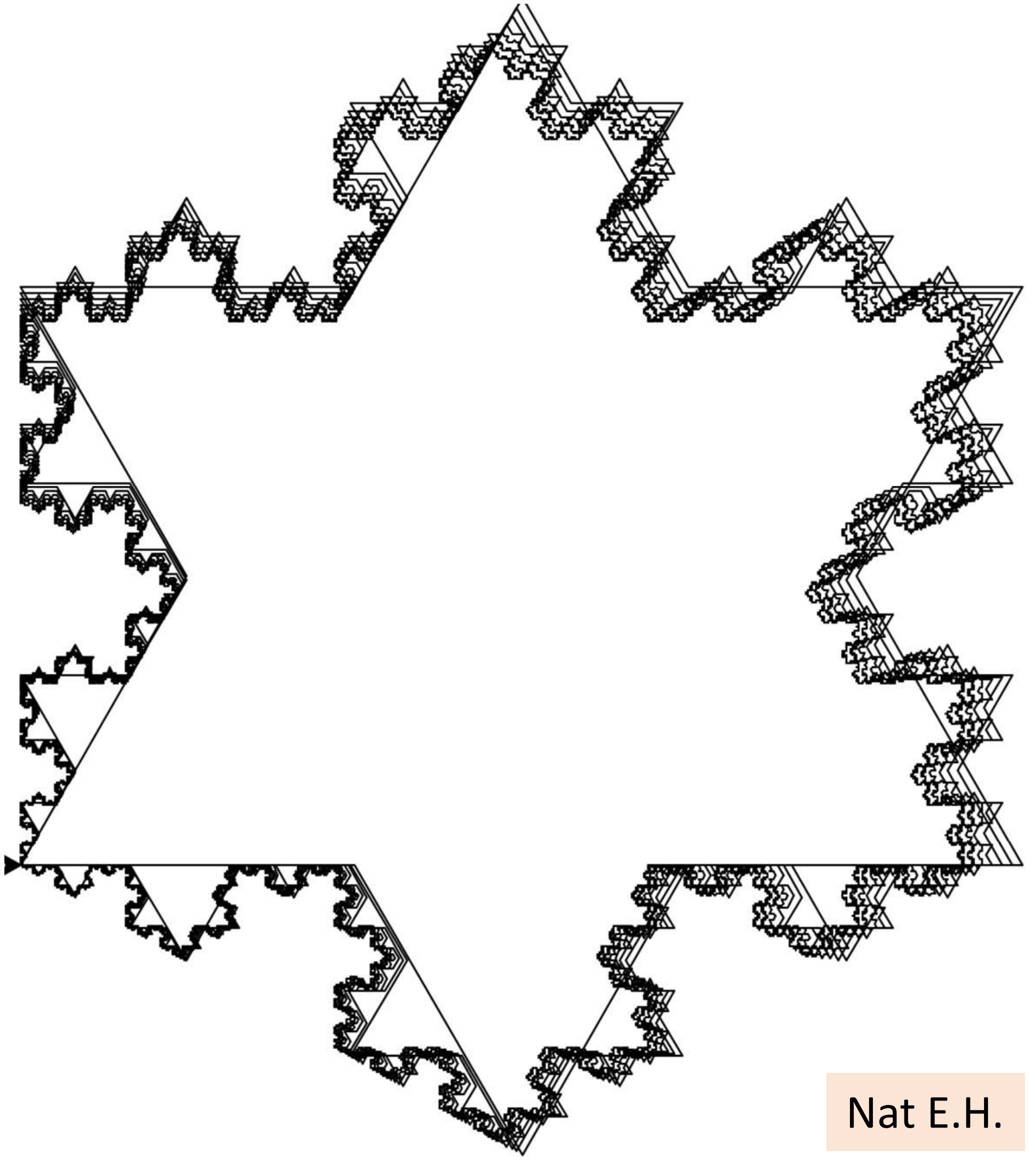
Fractals!



Emma Q



Hailey L

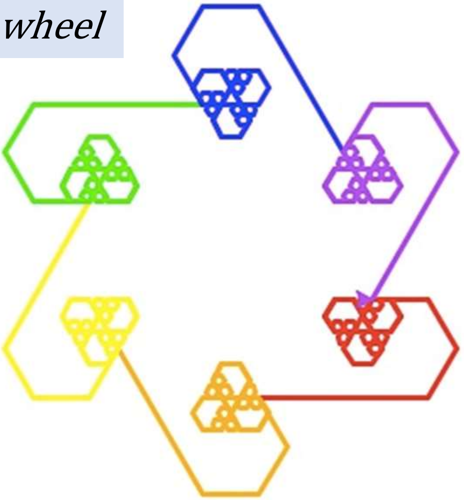


Nat E.H.

Well-named images!

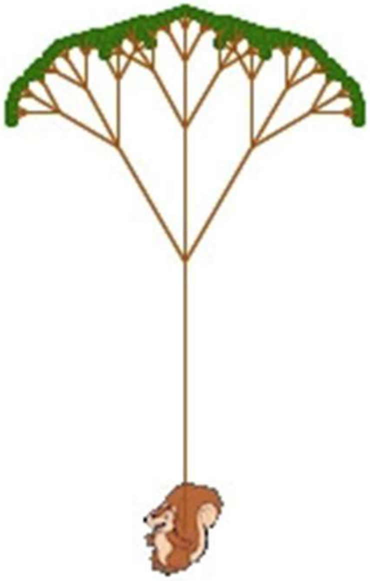
city_harbour

rainbow pinwheel

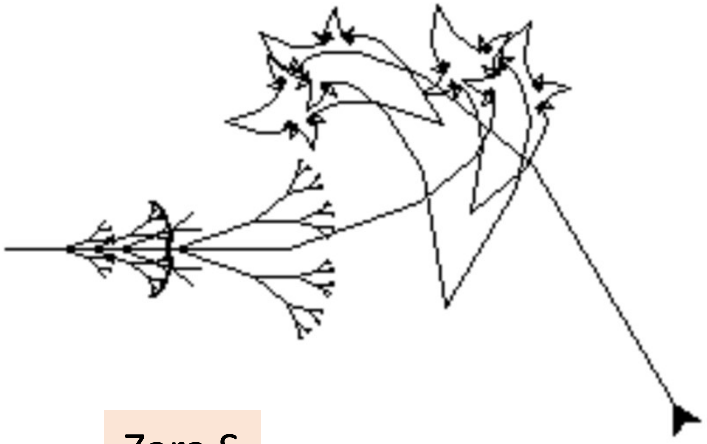


Roxy K.

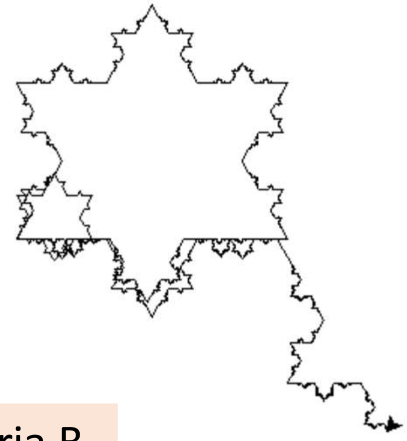
tree squirrel



dandelion_s

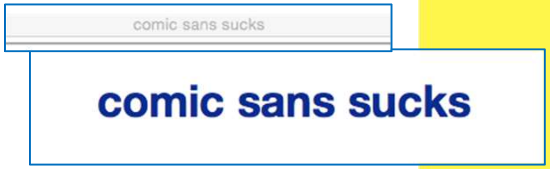


Zara S



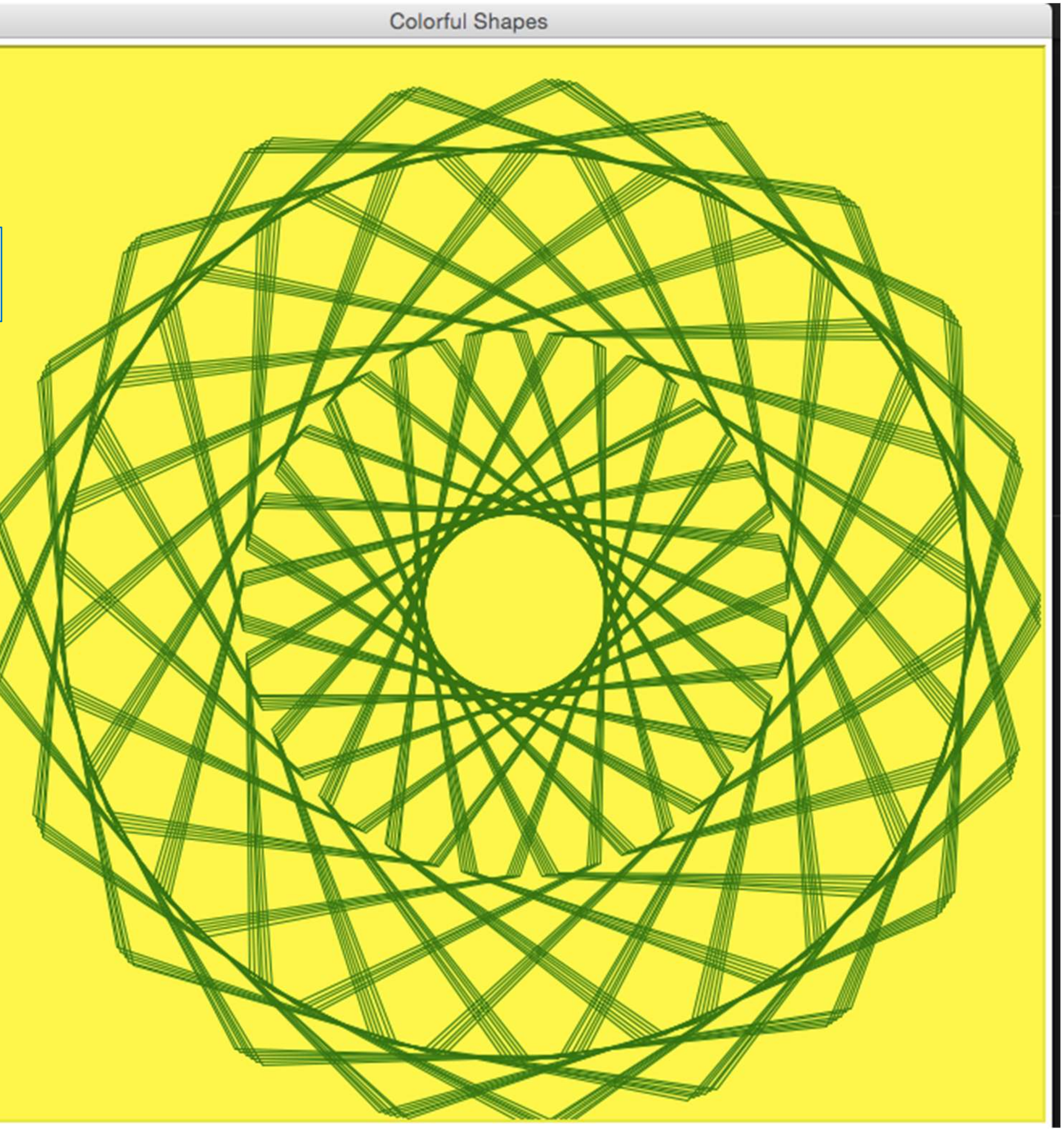
Gloria B.

star_kite



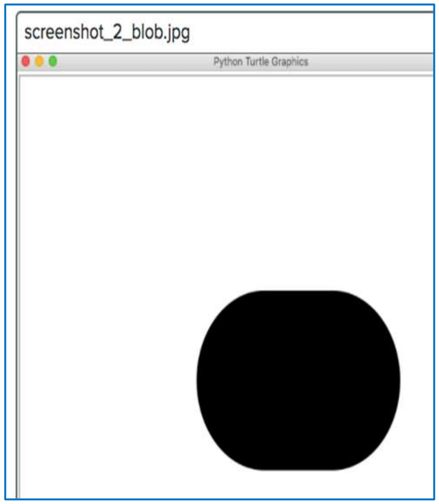
comic sans sucks

comic sans sucks

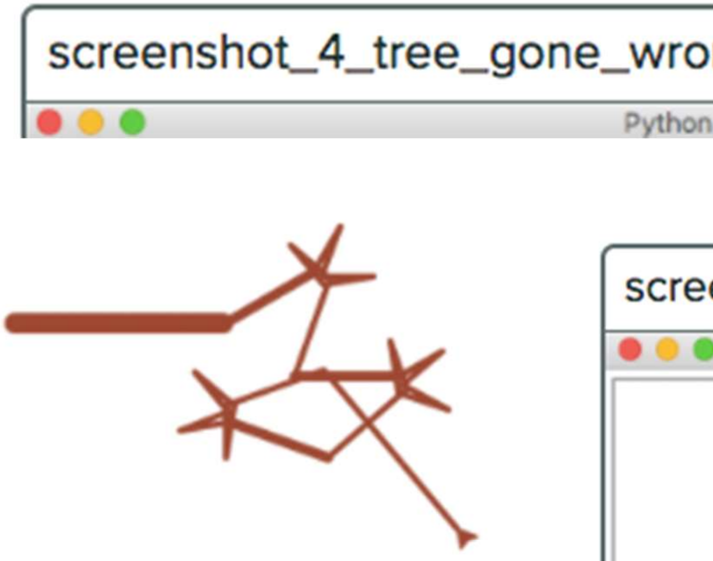


Dina R.

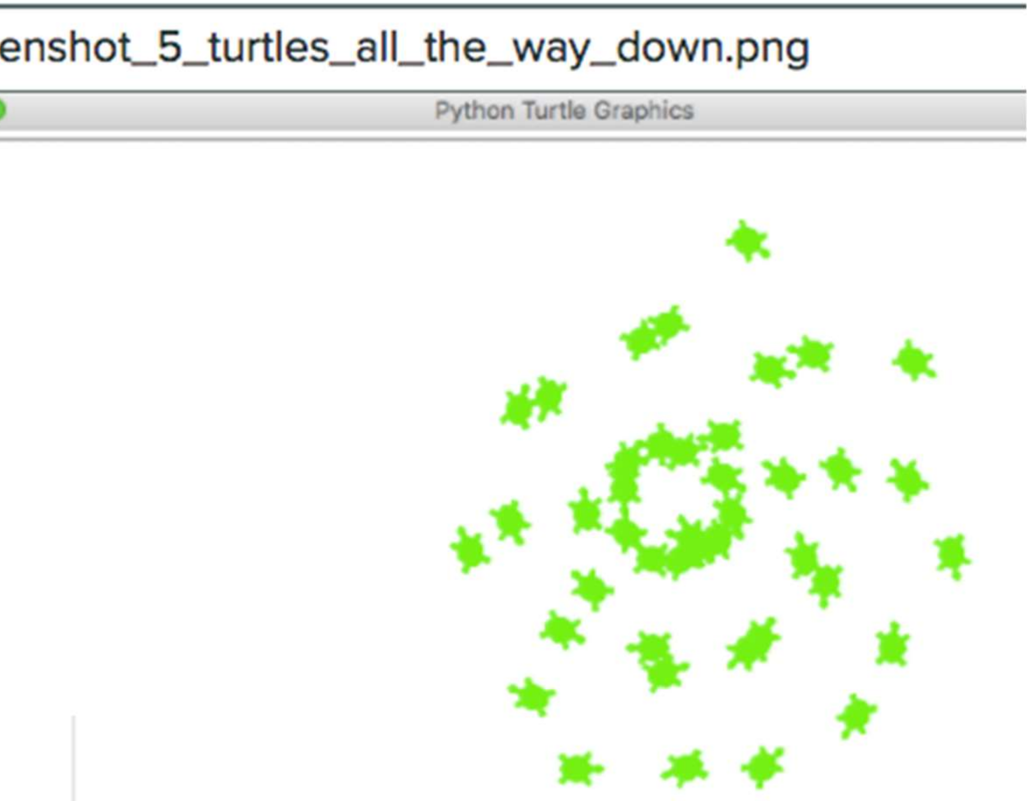
Joe. K



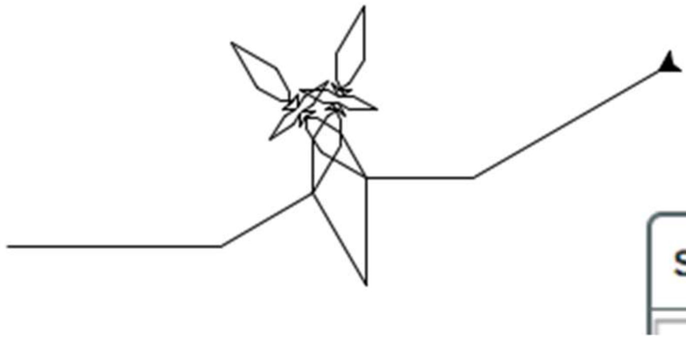
Veronica M



screenshot_4_tree_gone_wrong.png

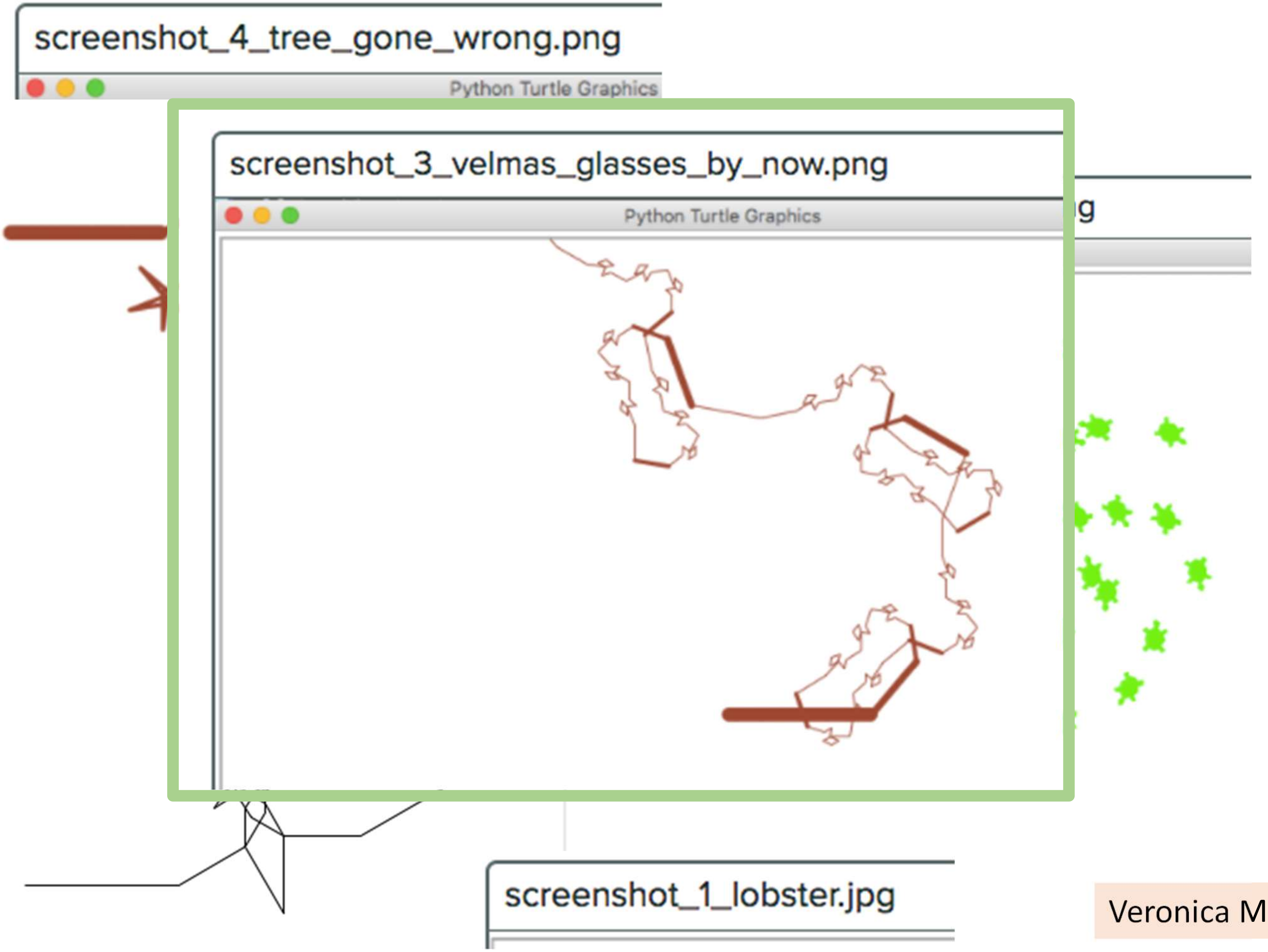


screenshot_5_turtles_all_the_way_down.png



screenshot_1_lobster.jpg

Veronica M



screenshot_4_tree_gone_wrong.png

screenshot_3_velmas_glasses_by_now.png

screenshot_1_lobster.jpg

Veronica M

badha

g

Jeni Z.

lie K.

i call this one, “kid on a leash on Halloween”

see?

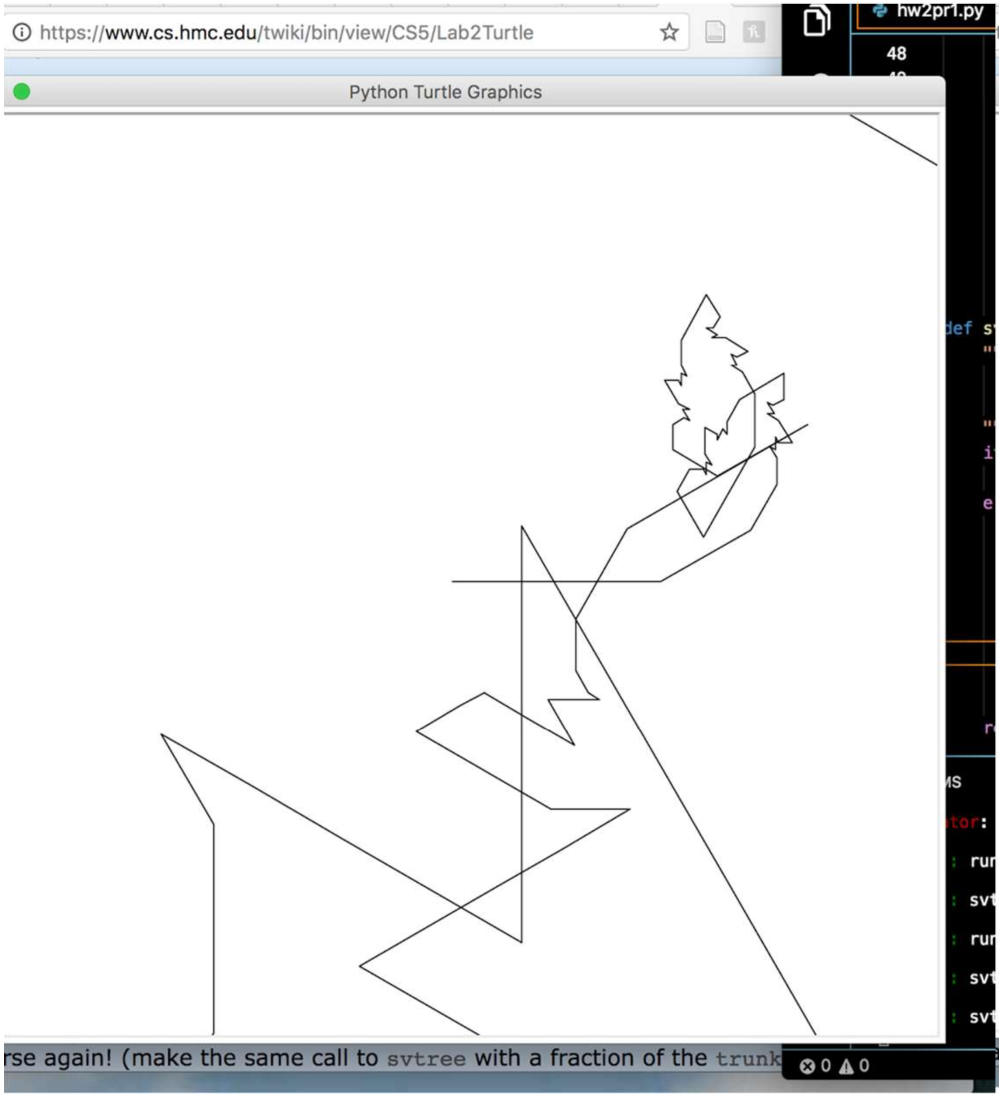
ill

ee.PNG

Pet

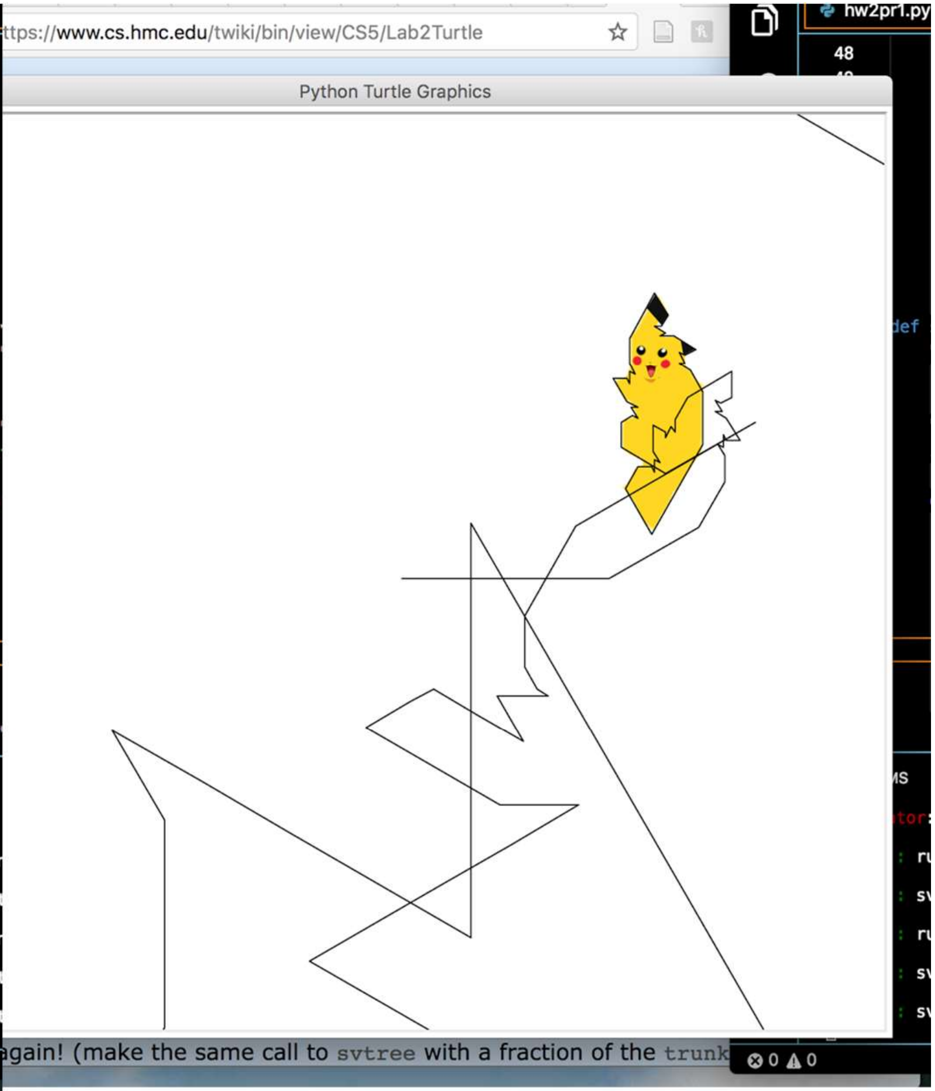
Vaiva P.

Please find enclosed my extra credit assignment from Lab. I have included the original screenshot of my failure, followed by a badly rendered Pikachu version.



https://www.cs.hmc.edu/twiki/bin/view/CS5/Lab2Turtle

rse again! (make the same call to svtree with a fraction of the trunk

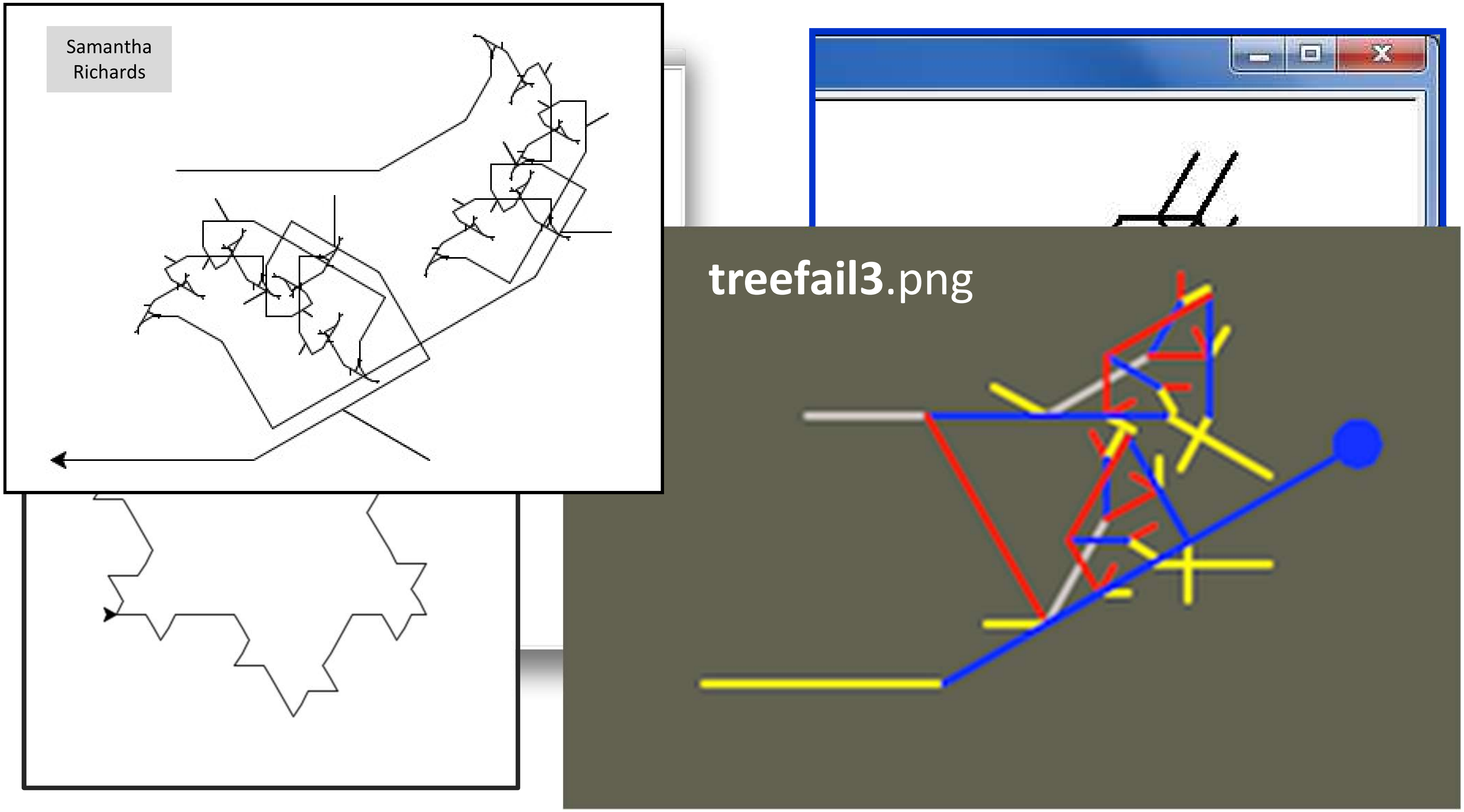


ttps://www.cs.hmc.edu/twiki/bin/view/CS5/Lab2Turtle

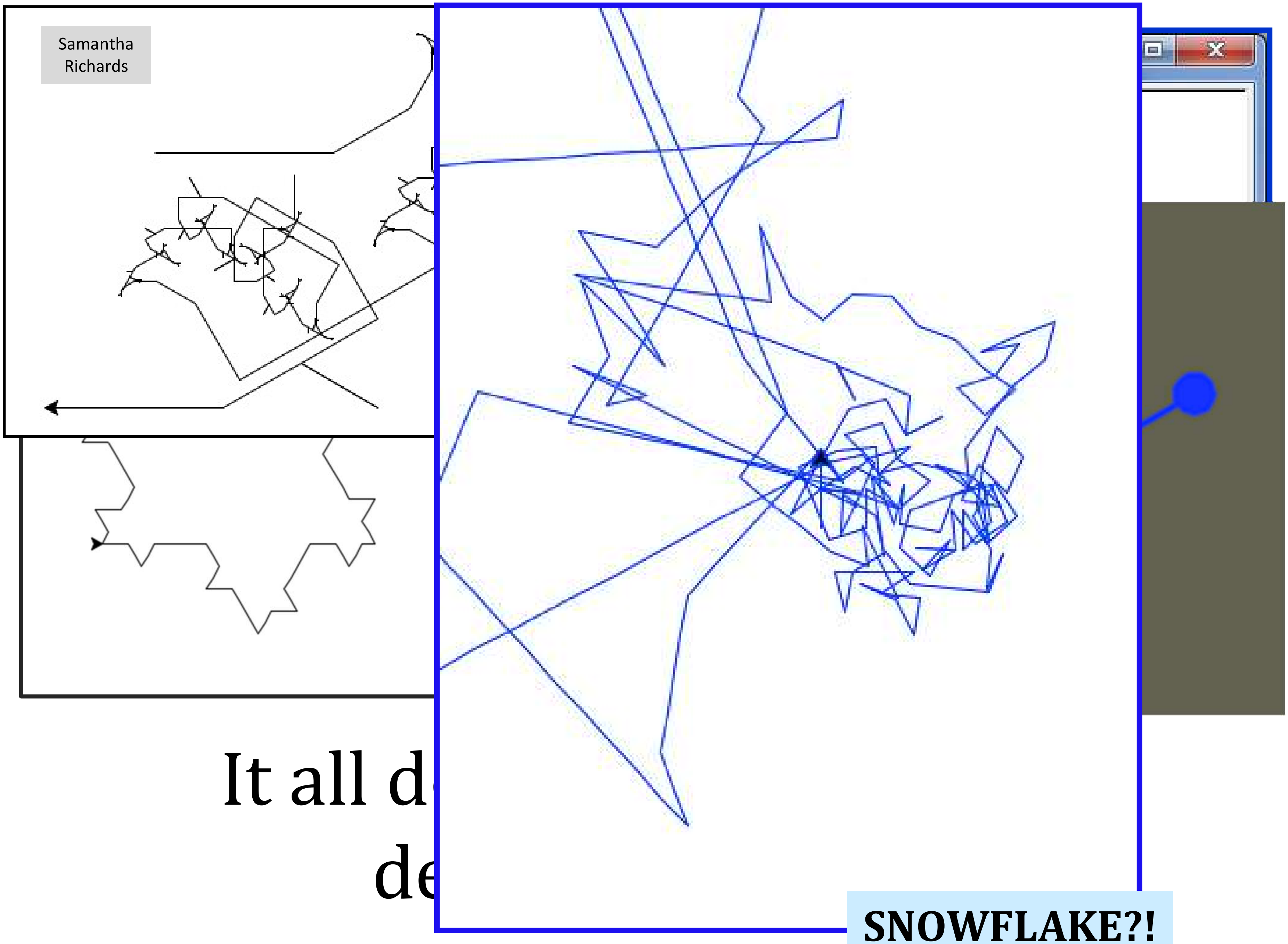
gain! (make the same call to svtree with a fraction of the trunk

Rachel L

It all depends on how you define ***success*** ...



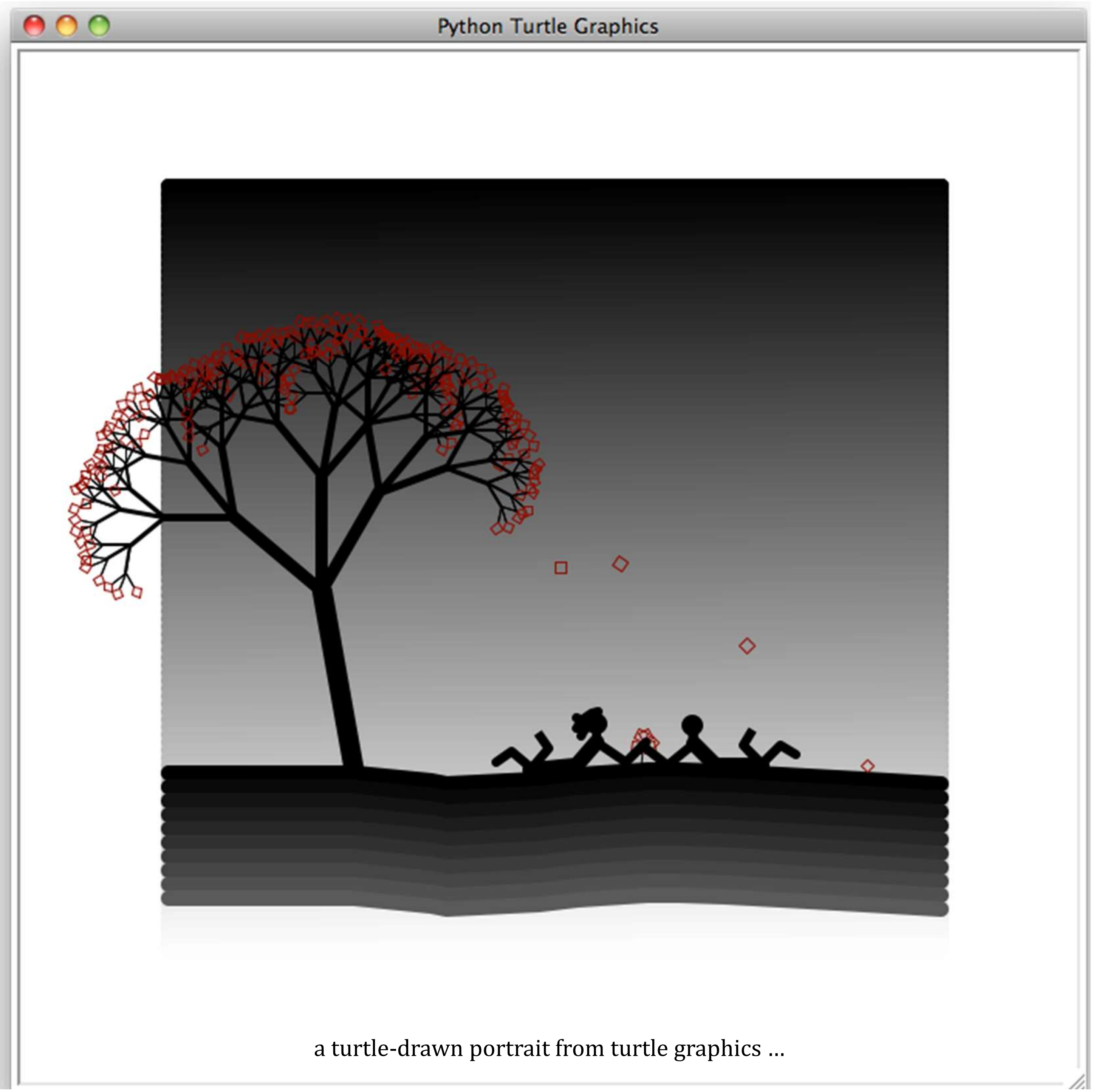
It all depends on how you define ***success*** ...



It all d

de

SNOWFLAKE?!



a turtle-drawn portrait from turtle graphics …

Whoa!
'12

Back to bits…

not the original name…

Lab 4: Computing in binary

base 2		base 10
… 4 2 1	=	100 10 1 141

This first step of **left-to-right** conversion into binary is tricky to program... ***Why?***

You mean ***aside*** from having to think in binary?

Lab 4: Computing in binary

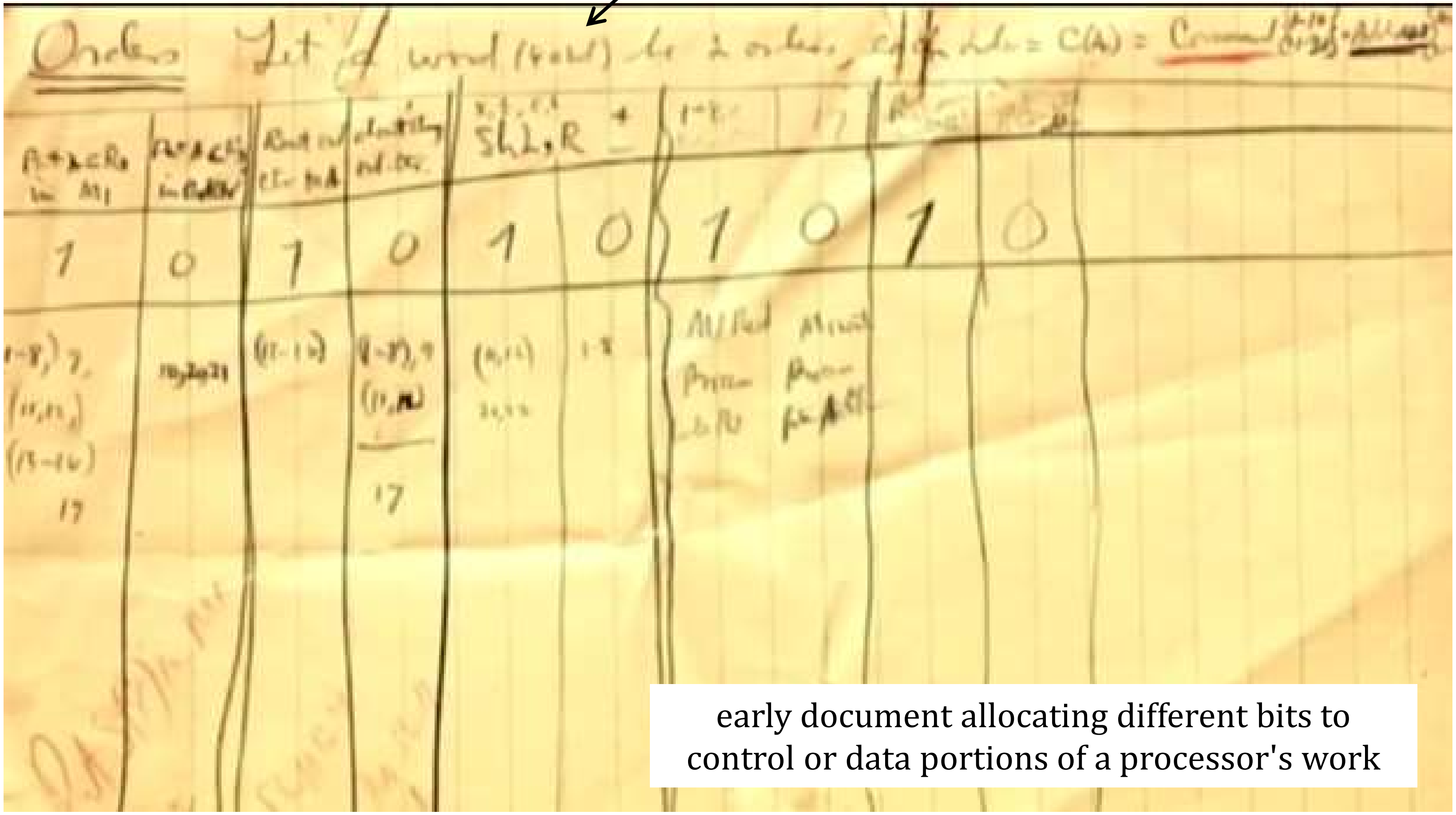
base 2		base 10
128 64 32 16 8 4 2 1	=	100 10 1 **141**

This first step of **left-to-right** conversion into binary is tricky to program... ***Why?***

It's tricky to find the largest power needed...

b.d. ~ binary digit ~ **bit**

"bit" first appeared in print in 1948 (Claude Shannon)


Orders
Sh L, R
1
0
1
0
1
0
1
0
1
0
17
17

early document allocating different bits to control or data portions of a processor's work

Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any earlier bits?* The last ***two***? ***three***?

All of them?

53

in the end,
we need
"53"-worth
of value

... **32s** **16s** **8s** **4s** **2s** **1s** top-level reality!

... 16s 8s 4s 2s 1s "next"-level reality...

... 8s 4s 2s 1s

... 4s 2s 1s

2s 1s

____ ____ ____ ____ ____ ____ bits!

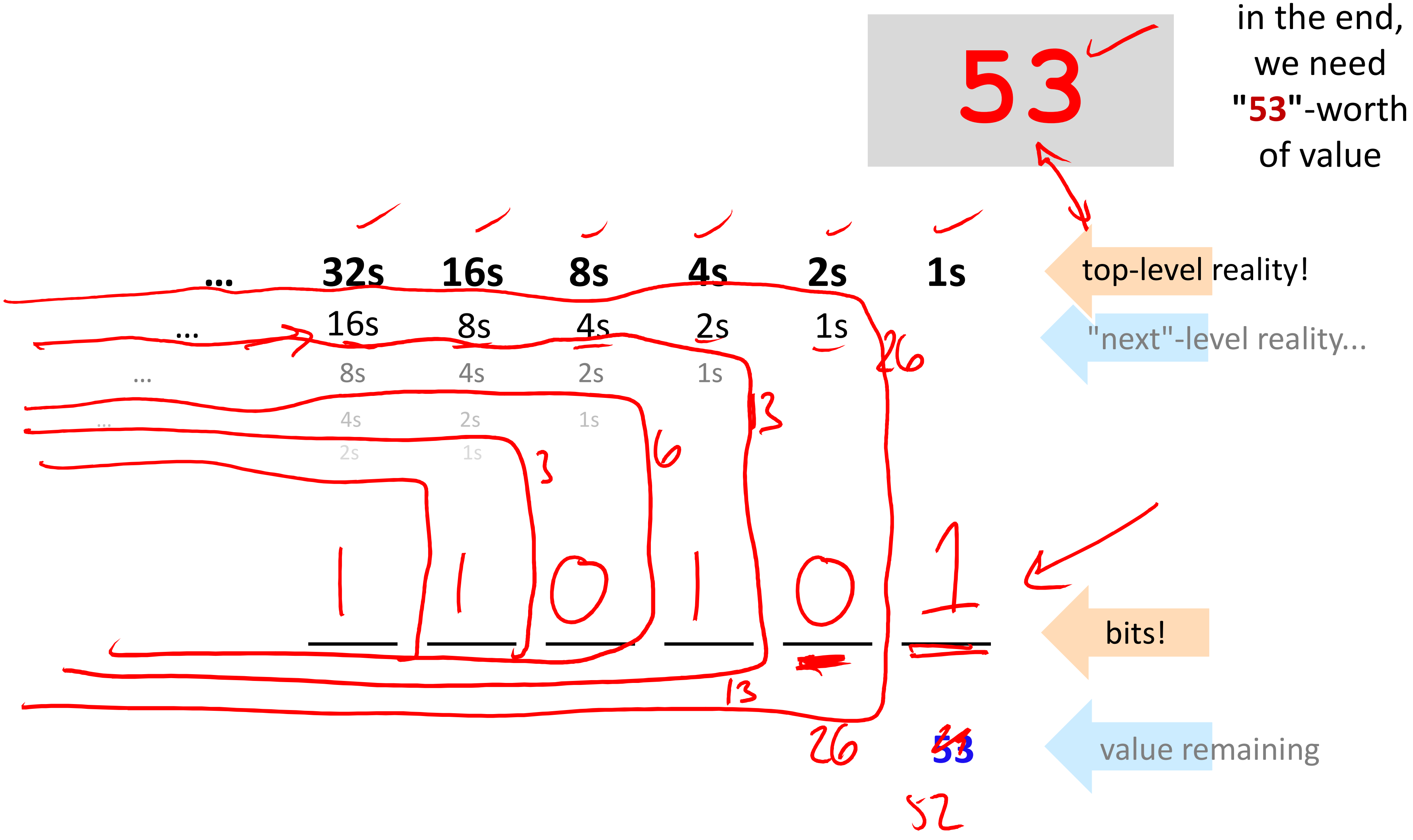
13

26

53 value remaining

Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any earlier bits?* The last ***two***? ***three***?

All *of them?*



53

in the end,
we need
"53"-worth
of value

...	32s	16s	8s	4s	2s	1s
...	16s	8s	4s	2s	1s	
...	8s	4s	2s	1s		
...	4s	2s	1s			
	2s	1s				

top-level reality!

"next"-level reality...

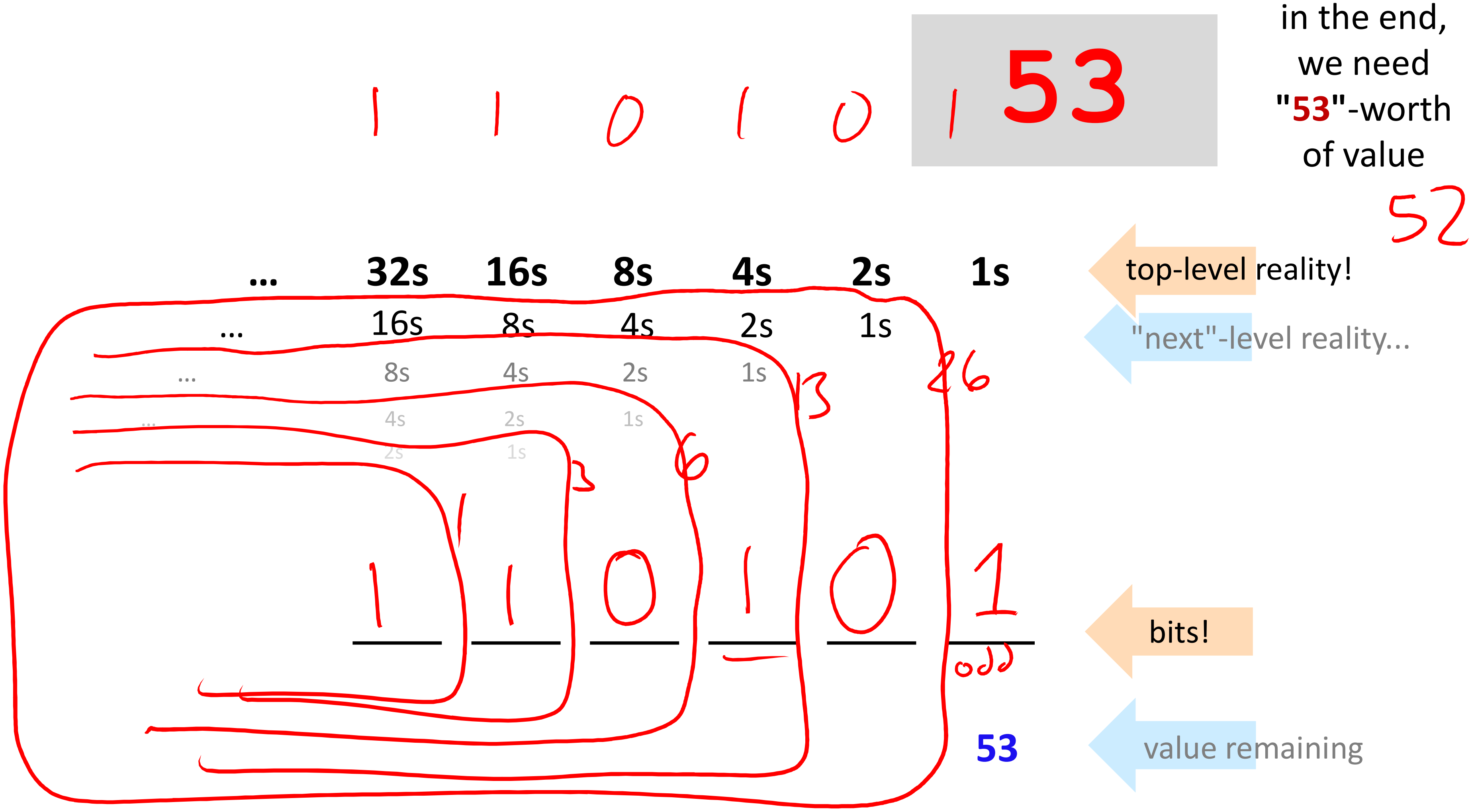
bits!

value remaining

53

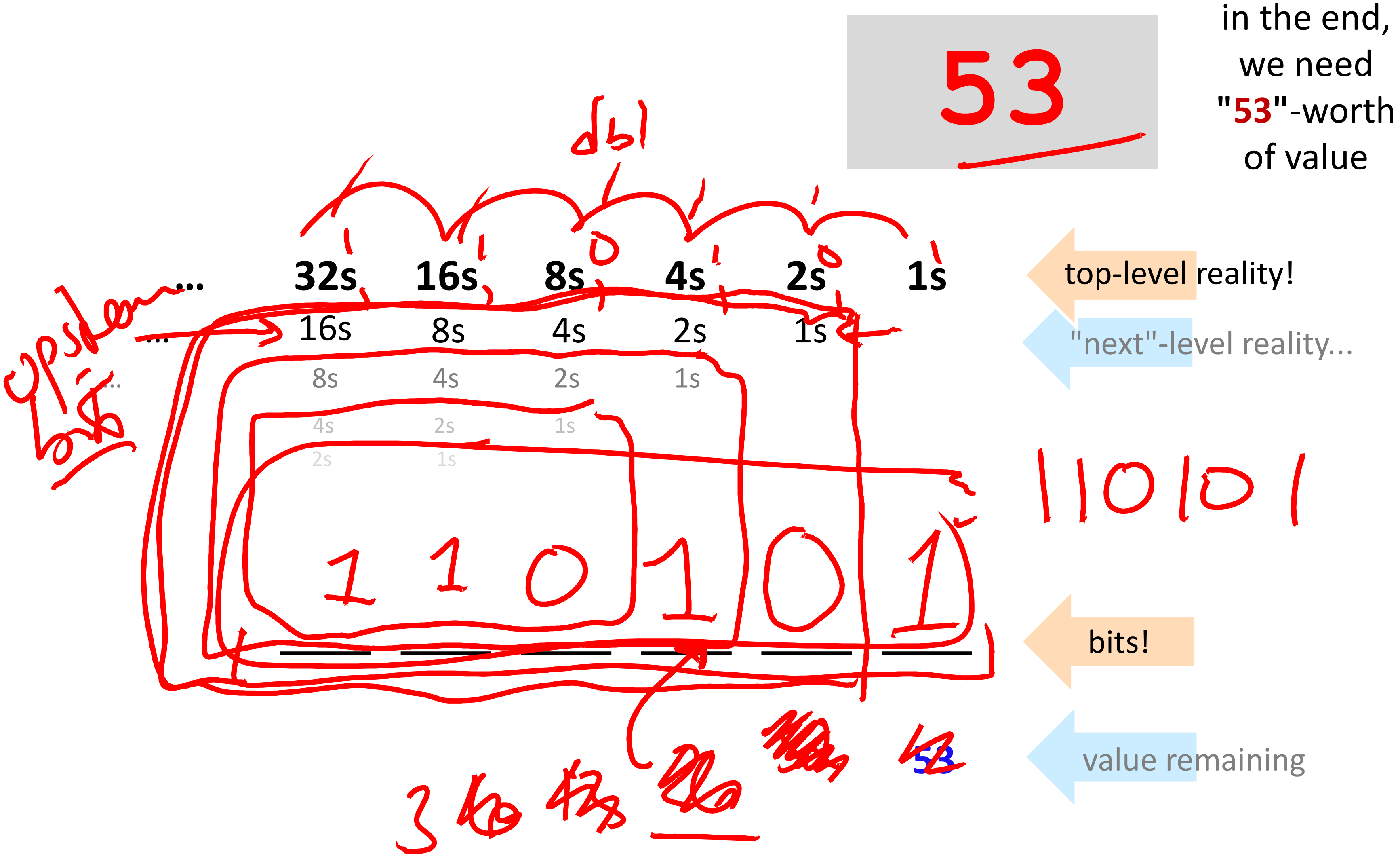
Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any earlier bits?* The last ***two***? ***three***?

All *of them?*



Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any earlier bits?* The last ***two***? ***three***?

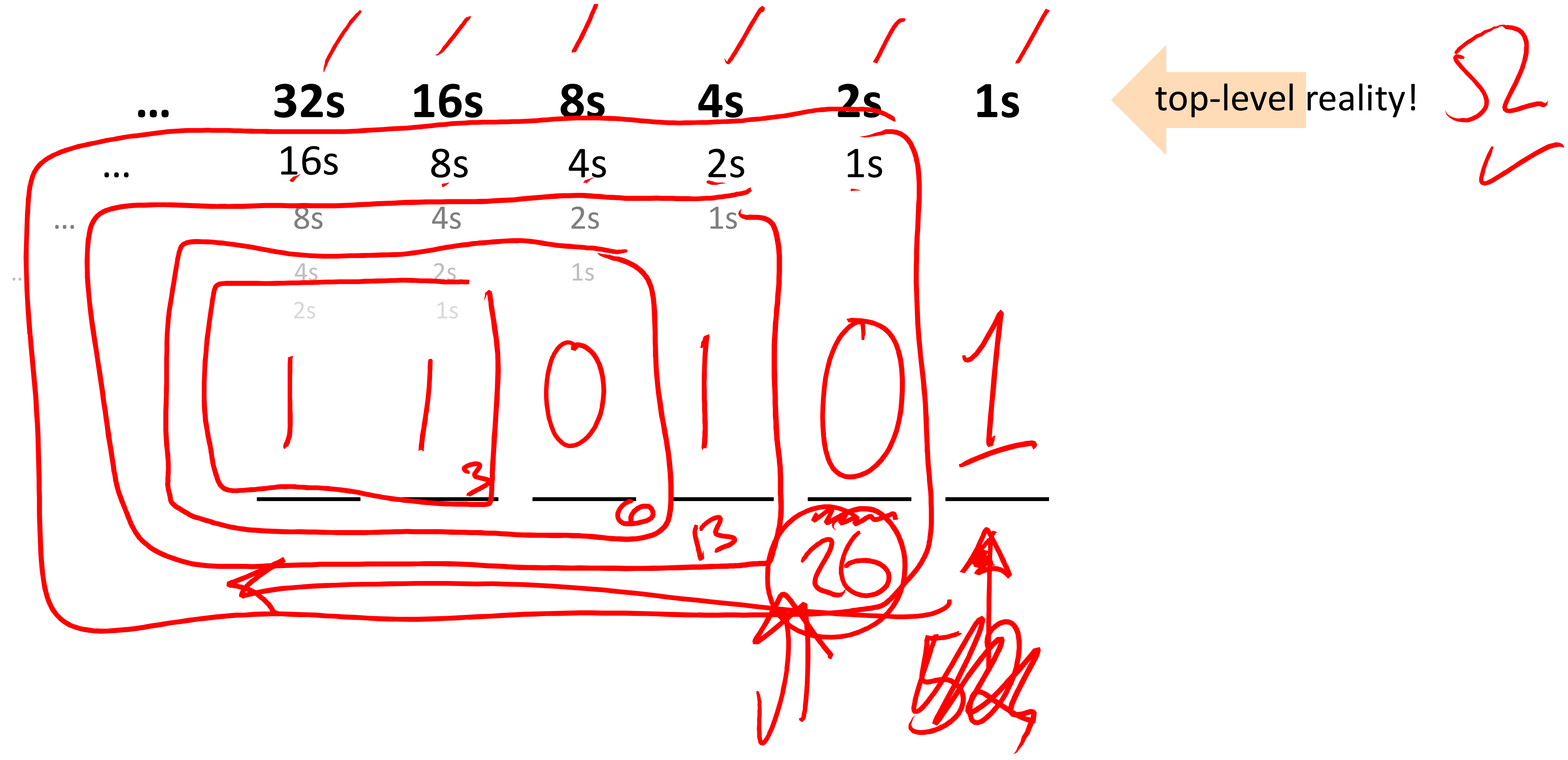
***All** of them?*



Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any earlier bits?* The last ***two***? ***three***?

All of them?

Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any other bits?* The last ***two***? ***3***? All?



... **32s** **16s** **8s** **4s** **2s** **1s**

... 16s 8s 4s 2s 1s

... 8s 4s 2s 1s

.. 4s 2s 1s

2s 1s

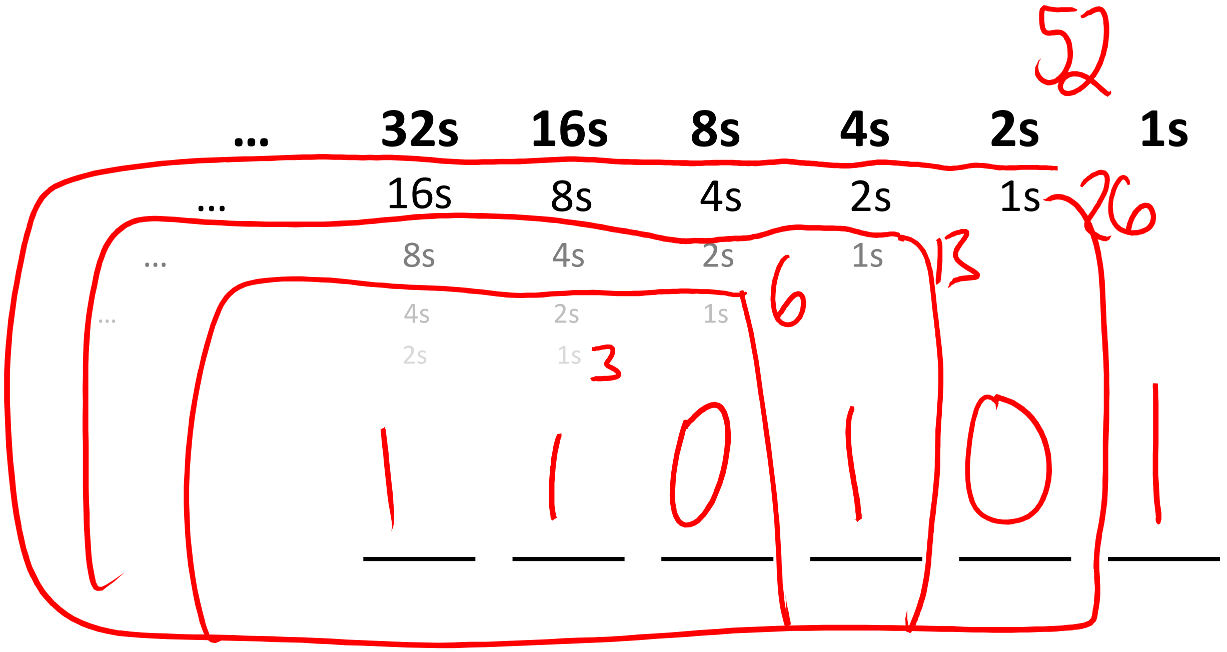
top-level reality!

53

in the end,
we need
"**53**"-worth
of value

Extra! Can you figure out the **<u>last binary digit</u>** (bit) of **53** *without determining any other bits?* The last ***<u>two</u>***? ***<u>3</u>***?

All?



… 32s 16s 8s 4s 2s 1s

… 16s 8s 4s 2s 1s

… 8s 4s 2s 1s

… 4s 2s 1s

2s 1s

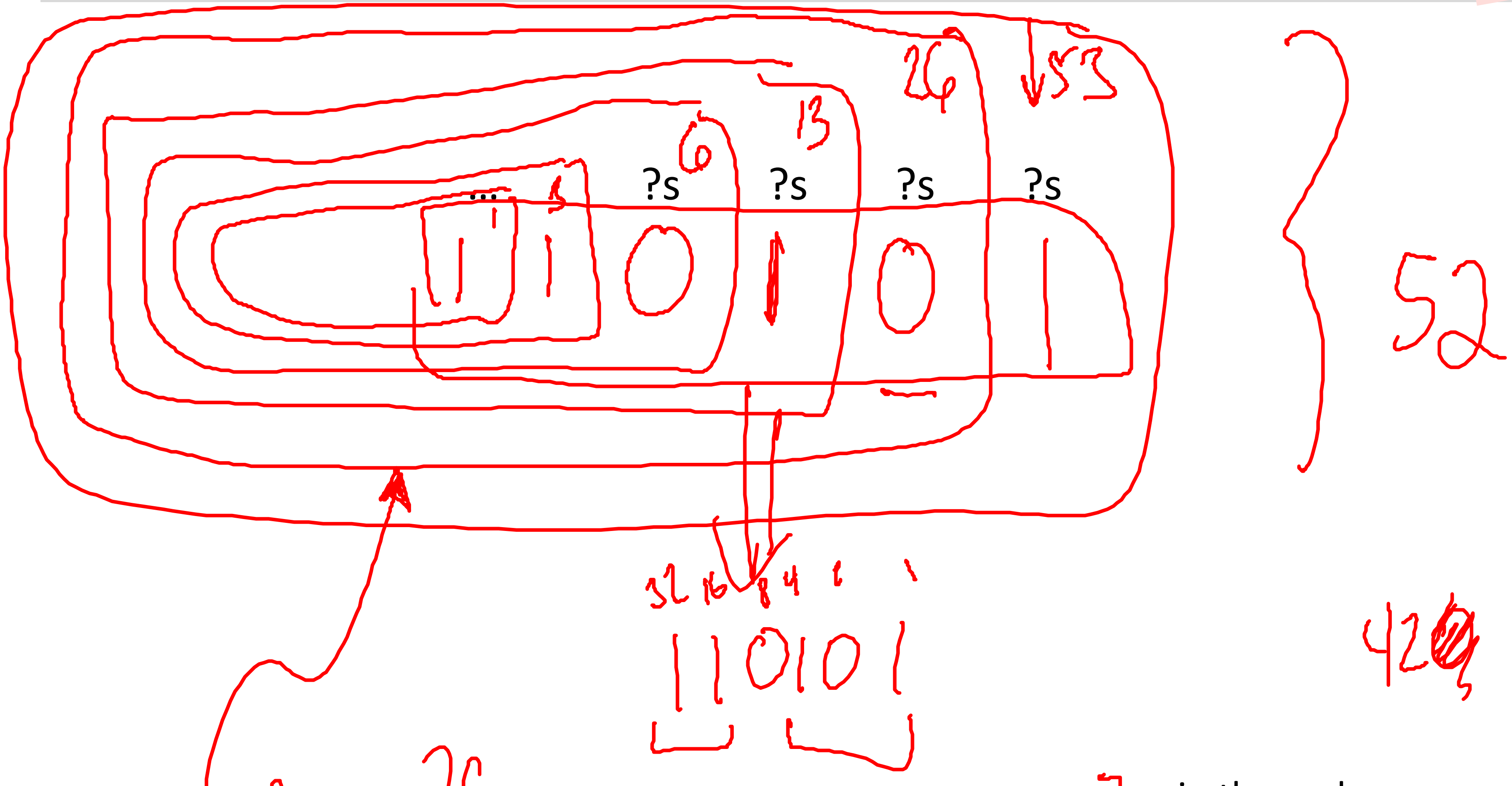
top-level reality!

53

in the end,
we need
"**53**"-worth
of value

Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any other bits*? The last two? three?

All?



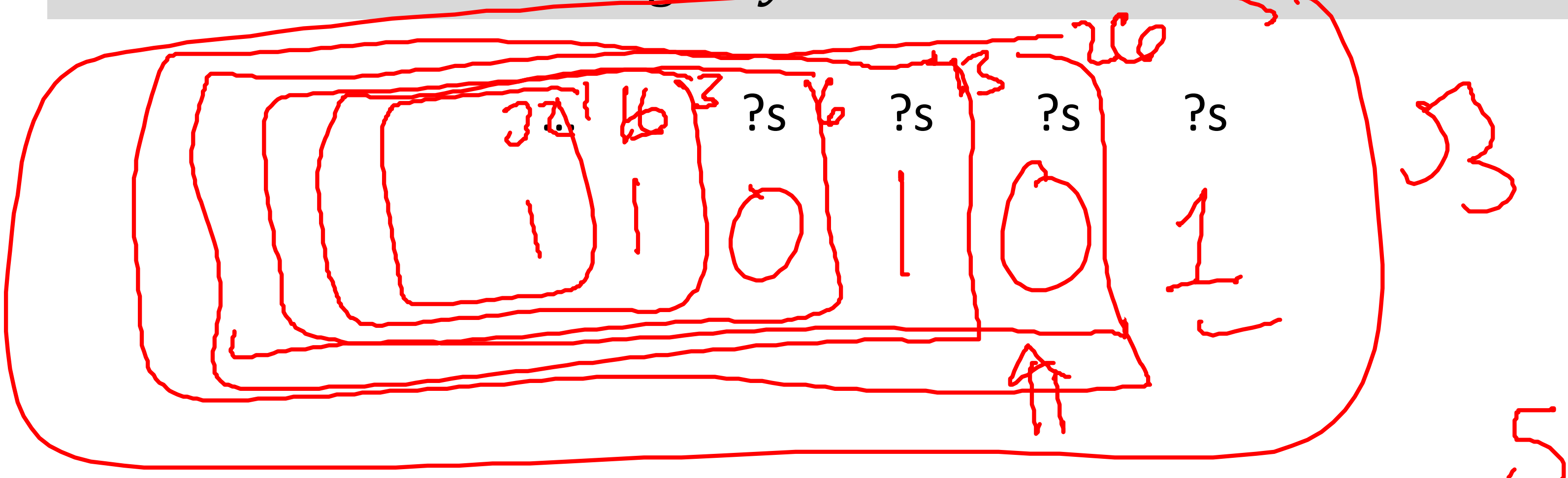
$52/2 = 26$

53

in the end,
we need
"**53**"-worth
of value

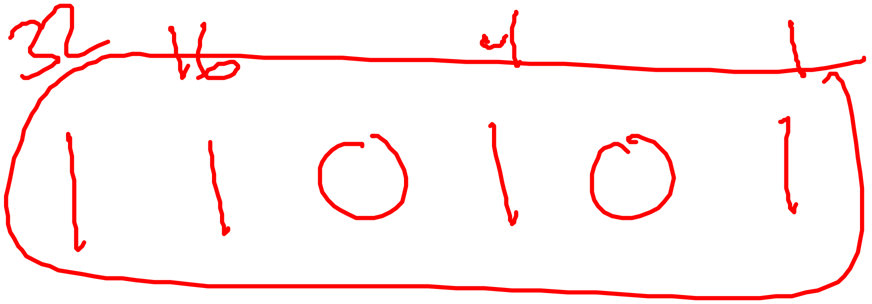
Extra! Can you figure out the **last binary digit** (bit) of **53** *without determining any other bits?* The last two? three?

All?



53 value

52 value remaining



53

in the end,
we need
"**53**"-worth
of value

Lab 4: Converting to binary...

base 10		base 2
100 10 1 141	=	128 64 32 16 8 4 2 1

What does the fact that 141 is ODD tell us?!

Try **right-to-left**!

141	=	10001101 128 64 32 16 8 4 2 1	answer

Lab 4: Converting to binary...

base 10		base 2
100 10 1		128 64 32 16 8 4 2 1
141	=	10001101

141/2

70/2

35/2

17/2

8/2

4/2

2/2

1

What does the fact that 141 is ODD tell us?!

Try **right-to-left**!

141	=	10001101
		128 64 32 16 8 4 2 1

answer

Lab 4: Computing in binary

base 10		base 2
100 10 1 **141**	=	128 64 32 16 8 4 2 1 **'10001101'** ←

Right-to-left works!

numTоBinary(N) → changed?

binaryToNum(S)

You'll write these right! (-to-left)

we need to *represent* binary numbers with **strings**

```
n2b(141)
```

```
b2n('10001101')
```

Lab 4: Computing in binary

base 10		base 2
100 10 1 141	=	128 64 32 16 8 4 2 1 '10001101'

```
def numToBinary( N ):
    if N == 0:
        return ''
    elif N%2 == 0:
        return  numToBinary(      ) +      
    else:
        return  numToBinary(      ) +      
```

empty string means 0

?

If N is even, what is the final bit?

How much VALUE is left to convert!?

If N is odd, what is the final bit?

Lab 4: ***Fleek*** binary conversion *!*

```
def numToBinary( N ):
    if N == 0: return ''
    else: return numToBinary(N//2) +
```

```
def numToBinary( N ):
    if N == 0:
        return ''
    elif N%2 == 0:
        return  numToBinary( N//2 )+ '0'
    else:
        return  numToBinary( N//2 )+ '1'
```

empty string means 0

If N is even, what is the final bit?

If N is odd, what is the final bit?

How much VALUE is left to convert!?

Lab 4: *Fleek* binary conversion *!*

```
def numToBinary( N ):
    if N == 0: return ''
    else: return numToBinary(N//2) +  str(N%2)
```

Use this page in lab today!!

```
def numToBinary( N
    i

    el
              numToBinary( N//2 )+ '0'
    else:
        return  numToBinary( N//2 )+ '1'
```

If N is even, what is the final bit?

If N is odd, what is the final bit?

How much VALUE is left to convert!?

Reasoning, *bit by bit*

<< left-shift

>> right-shift

& and

| or

and (both) & | or (either)

bitwise and

```
5:   101
6:   110
---------
&    100
```

5 & 6

4

bitwise or

```
5:   101
6:   110
---------
|    111
```

5 | 6

7

bitwise and

```
11: 1011
 5: 0101
---------
&
```

11 & 5

bitwise or

```
11: 1011
 5: 0101
---------
|
```

11 | 5

Reasoning, *bit by bit*

<<	>>	&	|
left-shift	right-shift	and	or

left-shift by 1

```
 11
110
```

```
3 << 1
```

6

What does ***left-shifting*** do to the **value** of a #?

left-shift by 2

```
  11
1100
```

```
3 << 2
```

12

right-shift by 1

```
101010
 10101
```

```
42 >> 1
```

21

What does ***right-shifting*** do to the **value** of a #?

```
42 >> 2
```

?

Being bit-*wise*

You **don't** need to convert to binary for these three...

`7 << 1`

left-shift

`5 << 4`

`170 >> 2`

right-shift

Try these for a bit...

```
14:   1110
 9:   1001
```

You ***do*** need to use binary for these two!

`14 | 9`

or

`14 & 9`

and

In processors shifts, ands, ors, adds, and subtractions are ***very fast***, whereas multiplying, dividing, and mod, which are relatively **slow**.

Given this, what is a way to compute these expressions using *only* ***fast*** operations, maybe in combination?

`N//4`

`N*7`

`N*17`

`N%16`

extra fleek!

Being bit-*wise*

You **don't** need to convert to binary for these three...

`7 << 1` ⇨ `14`

left-shift

`5 << 4` ⇨ `80`

`170 >> 2` ⇨ `42`

right-shift

Try these for a bit...

```
14:   1110
 9:   1001
```

You ***do*** need to use binary for these two!

`14 | 9` ⇨ `15` `1111`

or

`14 & 9` ⇨ `8` `1000`

and

In processors shifts, ands, ors, adds, and subtractions are ***very fast***, whereas multiplying, dividing, and mod, which are relatively **slow**.

Given this, what is a way to compute these expressions using *only* ***fast*** operations, maybe in combination?

`N//4`

`N*7`

`N*17`

`N%16`

extra fleek!

Let's first look at ***why*** you'd bother ... !?

Intel x86 processor instructions
and their speeds (2016)

In processors shift, and, or, add, and subtract are ***much faster*** than multiply, divide, and mod, which are ***relatively* slow.**

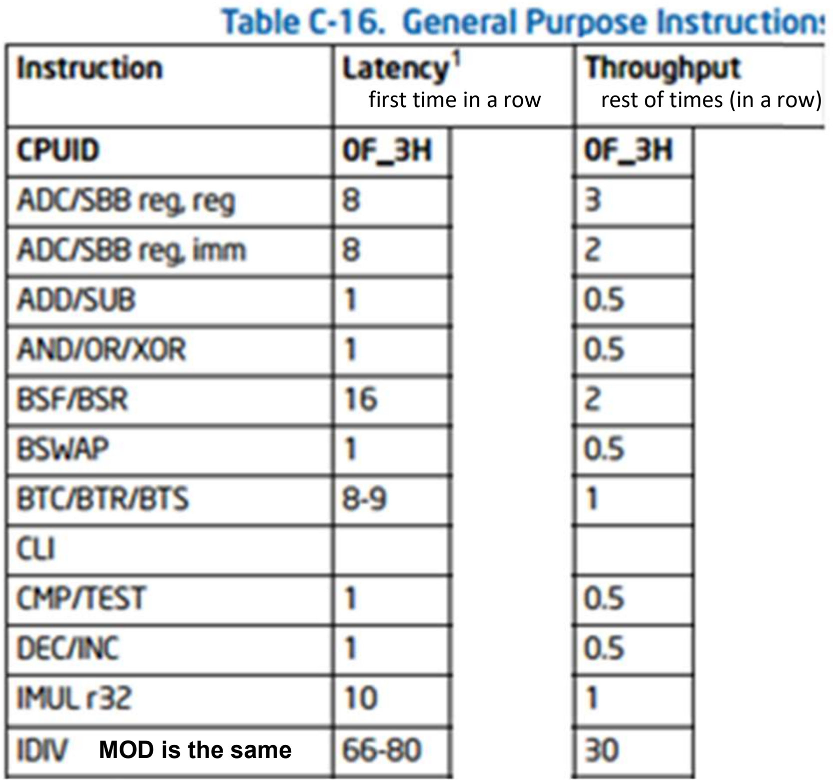
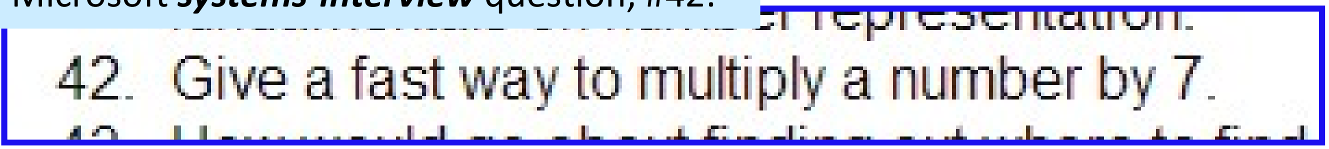


Table C-16. General Purpose Instructions

Instruction	Latency[1] first time in a row	Throughput rest of times (in a row)
CPUID	0F_3H	0F_3H
ADC/SBB reg, reg	8	3
ADC/SBB reg, imm	8	2
ADD/SUB	1	0.5
AND/OR/XOR	1	0.5
BSF/BSR	16	2
BSWAP	1	0.5
BTC/BTR/BTS	8-9	1
CLI		
CMP/TEST	1	0.5
DEC/INC	1	0.5
IMUL r32	10	1
IDIV MOD is the same	66-80	30

Intel® 64 and IA-32 Architectures

Old Microsoft ***systems-interview*** question, #42:



42. Give a fast way to multiply a number by 7.

Intel x86 processor instructions
and their speeds (2014)

In processors shift, and, or, add, and subtract are ***much faster*** than multiply, divide, and mod, which are ***relatively*** **slow.**

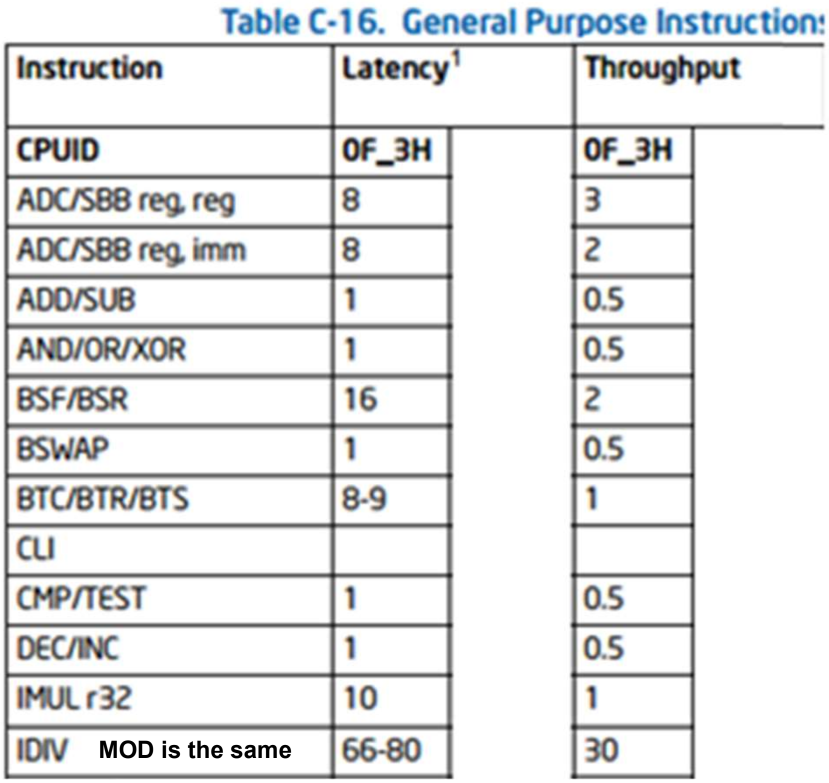


Table C-16. General Purpose Instruction

Instruction	Latency[1]	Throughput
CPUID	0F_3H	0F_3H
ADC/SBB reg, reg	8	3
ADC/SBB reg, imm	8	2
ADD/SUB	1	0.5
AND/OR/XOR	1	0.5
BSF/BSR	16	2
BSWAP	1	0.5
BTC/BTR/BTS	8-9	1
CLI		
CMP/TEST	1	0.5
DEC/INC	1	0.5
IMUL r32	10	1
IDIV MOD is the same	66-80	30

Given this, what is a way to compute these statements using combinations from <u>only</u> the ***fast*** operations above?

N//4 ⇨ N >> ____________

N*7 ⇨

N*17 ⇨

N%16 ⇨

Insight: Ancient Egyptian Multiplication

Ancient Egyptian multiplication

From Wikipedia, the free encyclopedia

Next time?

Insight: Ancient Egyptian Multiplication

```
21 × 6   ==   126
21    6
```

```
11 × 15   ==   165
11    15
```

AEM/RPM algorithm

Write the factors in two columns.

Repeatedly **halve** the LEFT and **double** the RIGHT. (toss remainders…)

Pull out the RIGHT values where the LEFT values are ***odd***.

Sum those values for the answer!

Why does this work?

Try it here

or *RPM…*

When traveling,
always insist on
binary
accommodations ... !

See you at lab
– *in just a* ***bit!***

This room is a 10!

Insight: Multiplication in binary w/o bits!

21	6
10	12
5	24
2	48
1	96

Decimal

⇒

$$\begin{array}{lrlr}
 & 110_{6} & & \\
\times & 10101_{21} & & \\
\hline
 & 110_{6} & & 6 \\
 & 0000_{12} & & \\
 & 11000_{24} & & 24 \\
 & 000000_{48} & & \\
+ & 1100000_{96} & + & 96 \\
\hline
 & 1111110_{126} & & 126
\end{array}$$

Binary

Insight: Multiplication in binary w/o bits!

11	15
5	30
2	60
1	120

Decimal

⇨

```
         1111 15
x        1011 11
  -------------
         1111 15          15
        11110 30          30
       000000 60
+     1111000 120       +120
  -------------       ------
     10100101 165        165
```

Binary