PIET -
An Artistic Programming Language

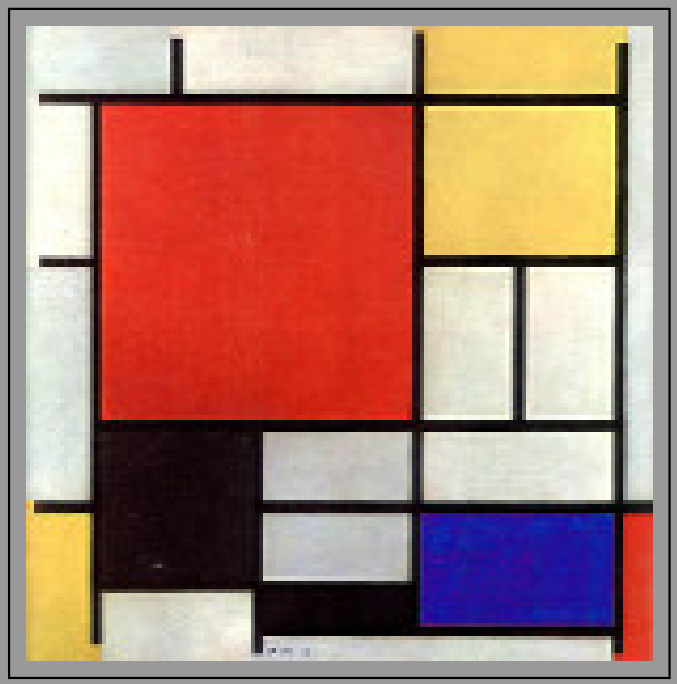
Manfred Moosleitner
Supervisor: Michael Färber

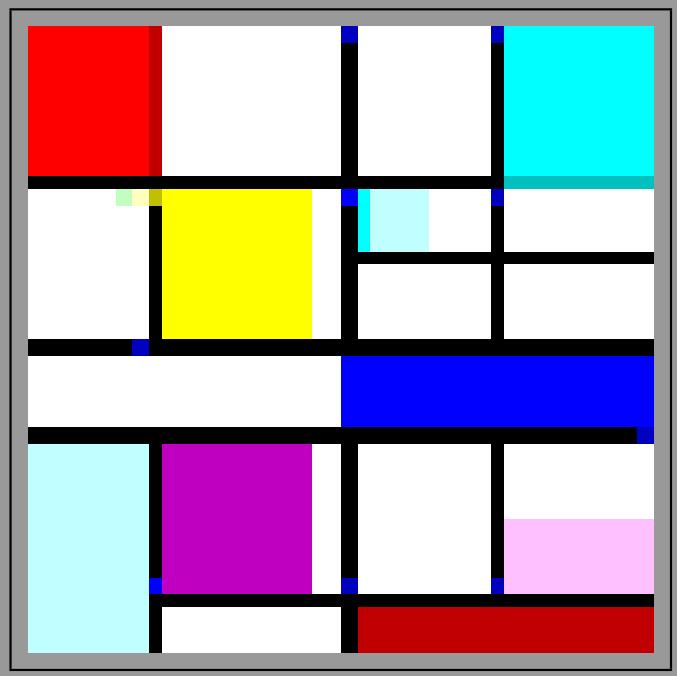
University of Innsbruck

2015-06-12

Warm Up!

Quiz: What are the two images below?





Warm Up!

Quiz: What are the two images below?

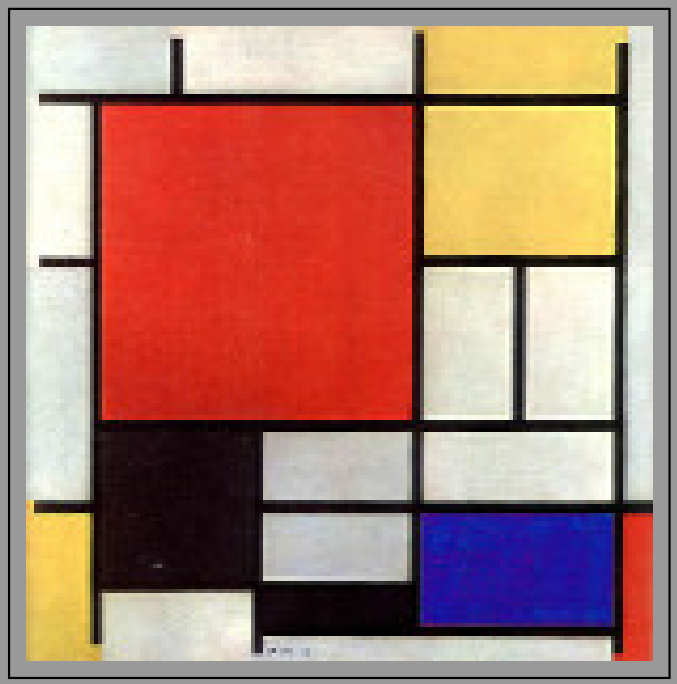


Figure: Composition with Red, Yellow, Blue and Black, by Piet Mondrian, 1926.[wikimedia]

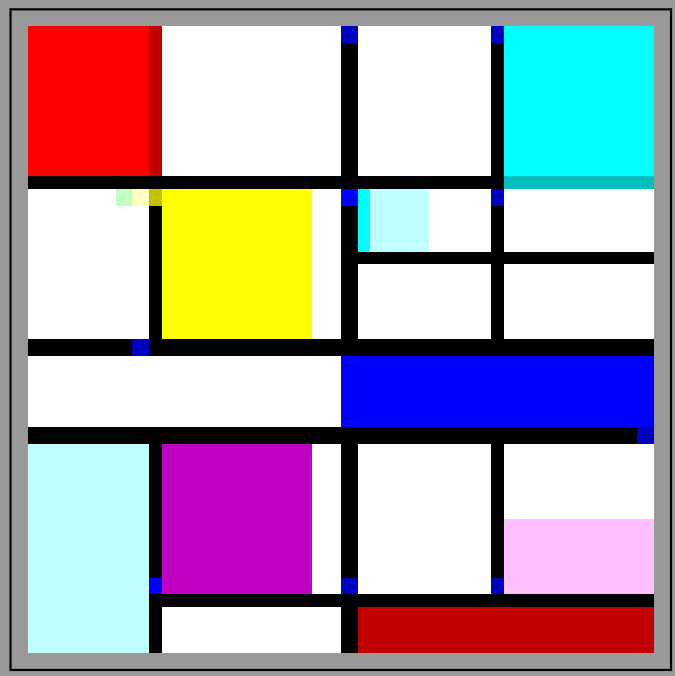


Figure: Example program in Piet, writes "Piet" as output. [3]

Welcome to the World of Piet!

Figure: Piet program to print “Hello, world!”.

Overview

We will take a look at...

Overview

We will take a look at...

- ...WHAT is Piet and WHO invented it.

Overview

We will take a look at...

- ...WHAT is Piet and WHO invented it.
- ...WHAT is Piet's specialty and HOW does Piet work.

Overview

We will take a look at...

- ...WHAT is Piet and WHO invented it.
- ...WHAT is Piet's specialty and HOW does Piet work.
- ...WHICH tools exist to create Piet programs.

Overview

We will take a look at...

- ...WHAT is Piet and WHO invented it.
- ...WHAT is Piet's specialty and HOW does Piet work.
- ...WHICH tools exist to create Piet programs.
- ...WHAT can Piet programs do.

Overview

We will take a look at...

- ...WHAT is Piet and WHO invented it.
- ...WHAT is Piet's specialty and HOW does Piet work.
- ...WHICH tools exist to create Piet programs.
- ...WHAT can Piet programs do.
- ...WHO is using Piet.

Author

David Morgan-Mar

- Scientist from Australia (astrophysics)
- Lecturer and teacher
- Role playing enthusiast
- Inventor of Piet and other esoteric languages (e.g. Chef, Ook, Zombie, ...)
- Created Piet in early 2000's

Vision and Eponym

Goal and Vision

"Piet is a programming language in which programs look like abstract paintings. The language is named after Piet Mondrian, who pioneered the field of geometric abstract art." [3]

Piet Mondrian

- Dutch artist (1872-1944)
- Founder of "Neoplastizismus" [5]

Figure: Mondrian in 1899 [wikimedia]

Execution Trace

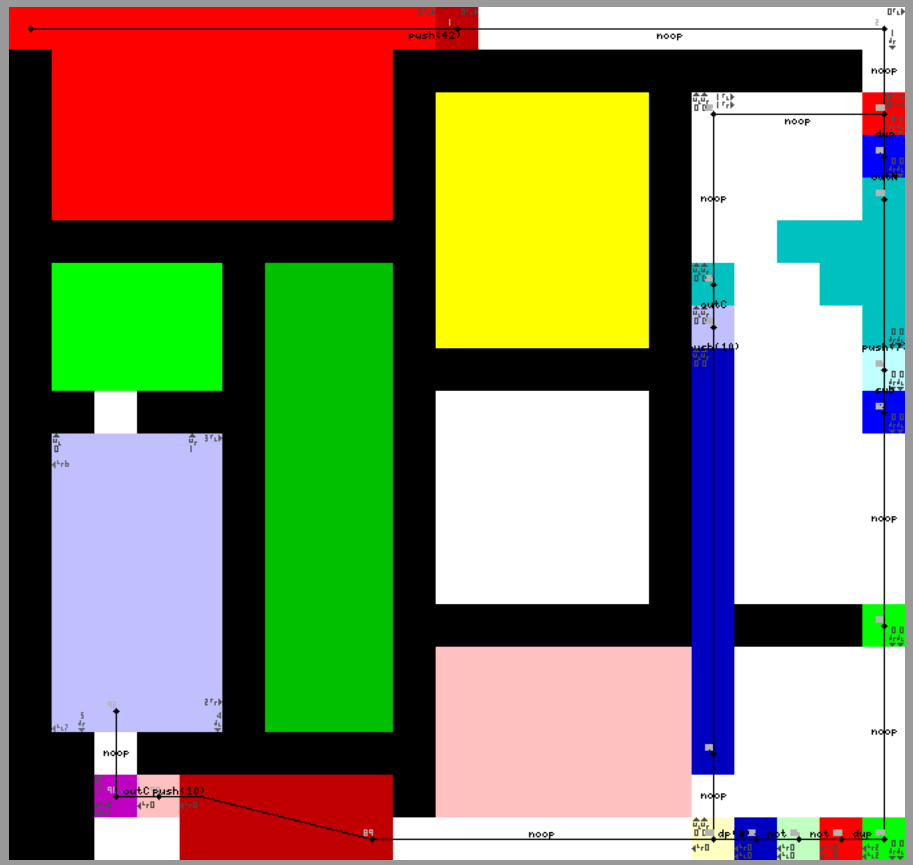
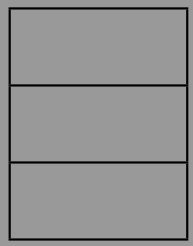


Figure: Trace of a Piet program.



Stack

command:

output:

Execution Trace

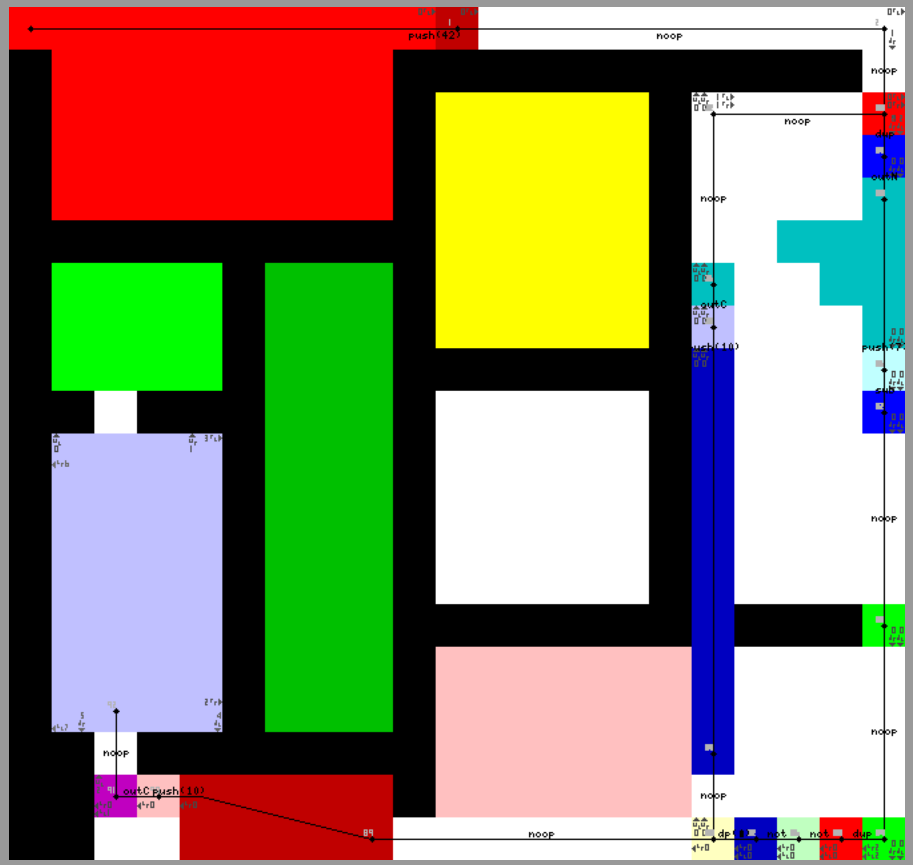


Figure: Trace of a Piet program.

Stack

command:
push(42)

output:

Execution Trace

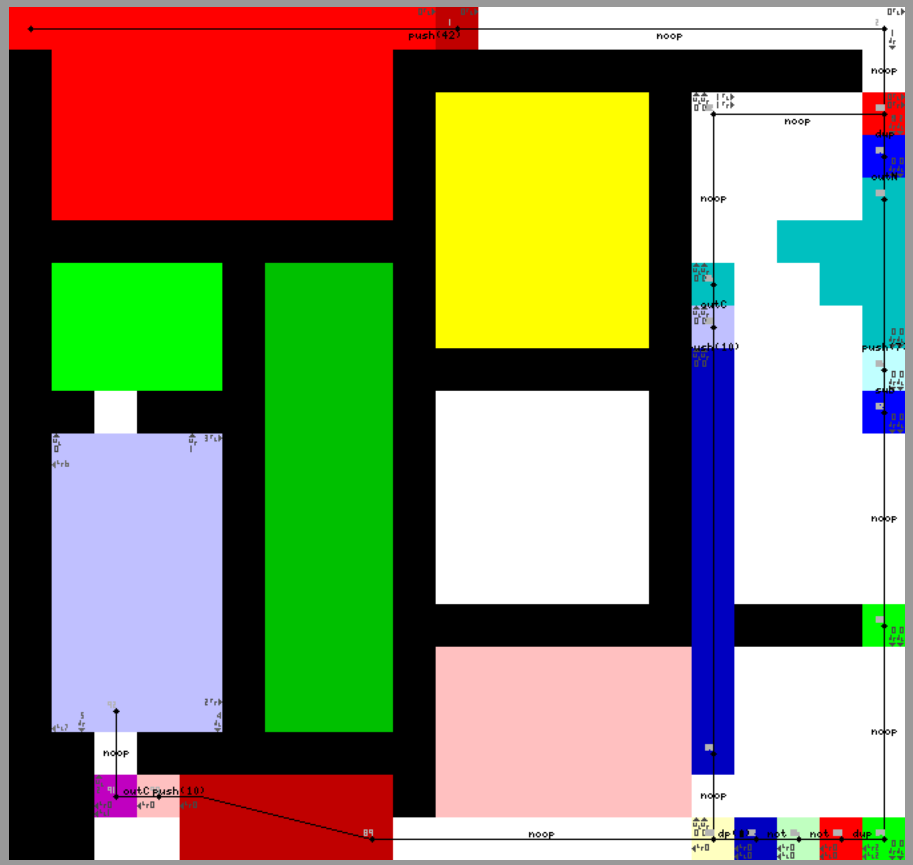
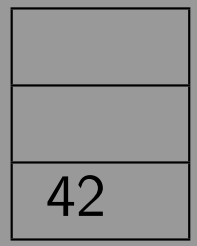


Figure: Trace of a Piet program.



Stack

command:
dup()

output:

Execution Trace

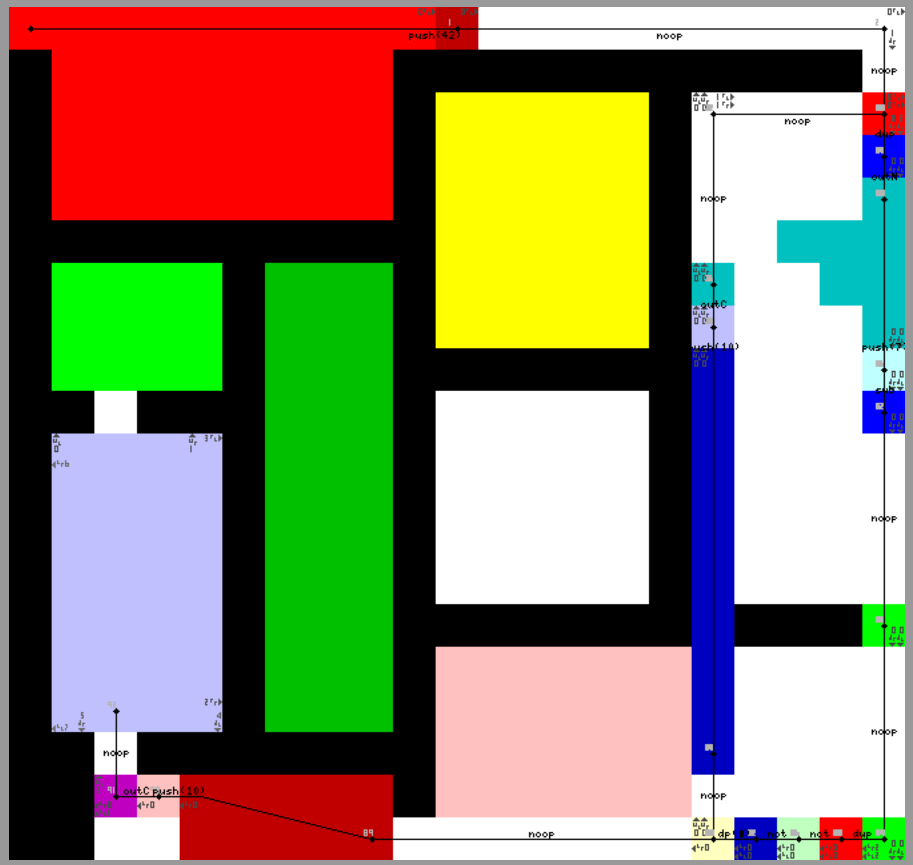


Figure: Trace of a Piet program.

42
42

Stack

command:
outN()

output:

Execution Trace

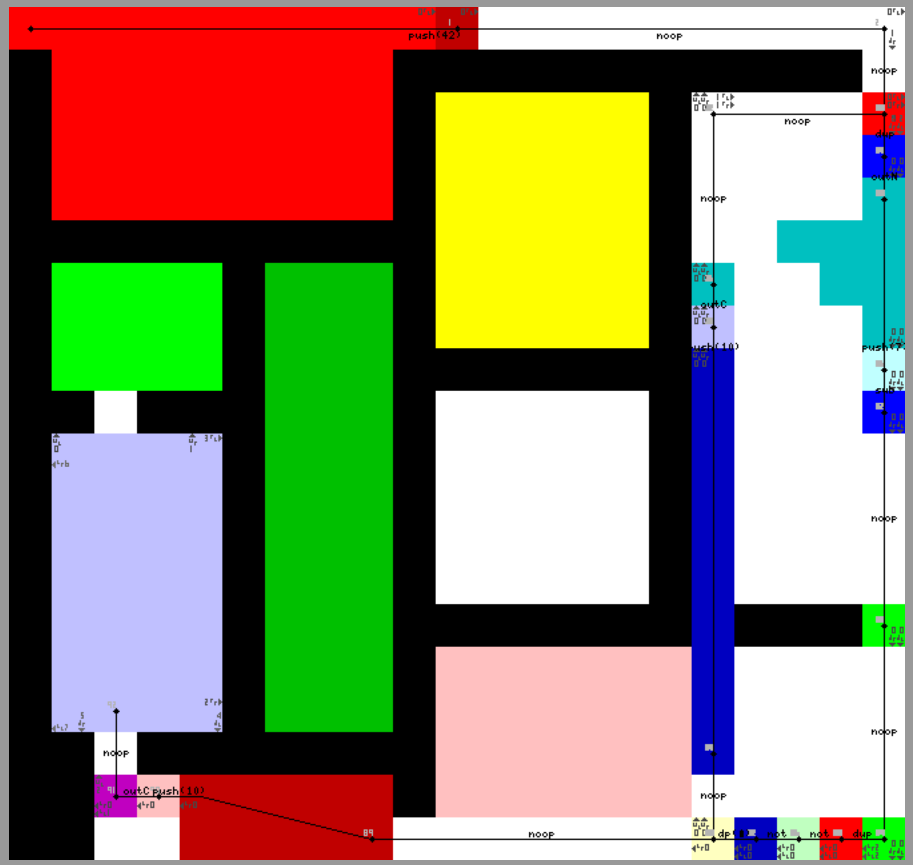
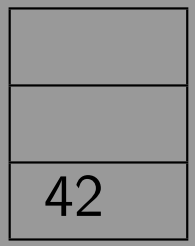


Figure: Trace of a Piet program.



Stack

command:

output:
42

Execution Trace

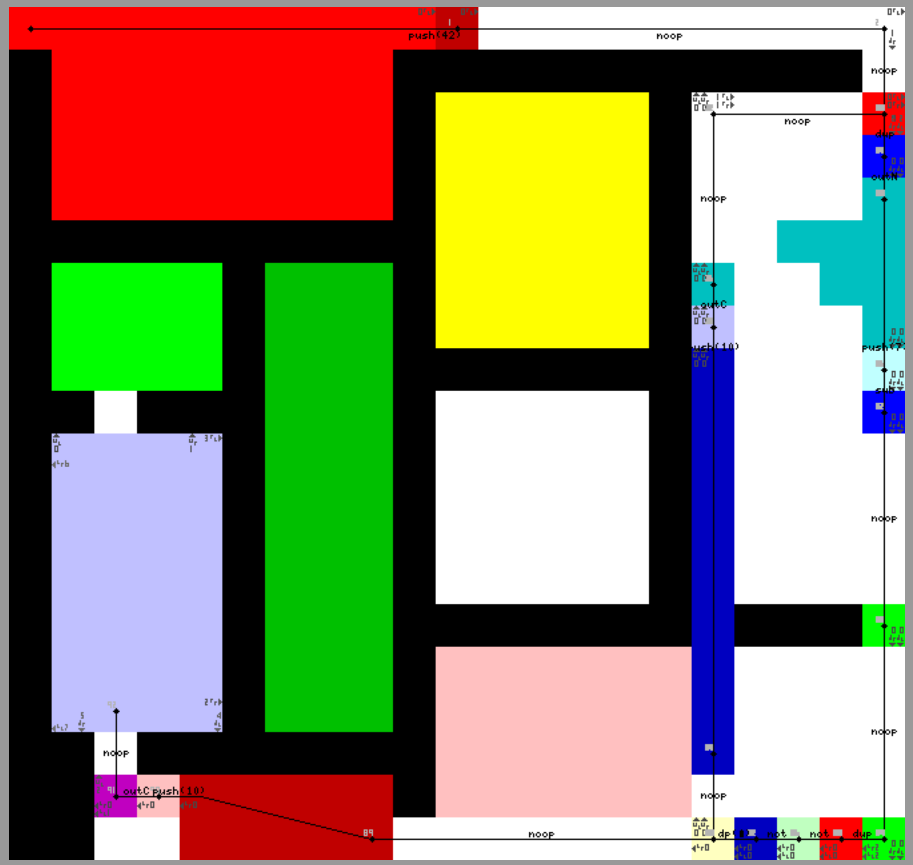
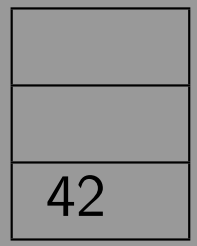


Figure: Trace of a Piet program.

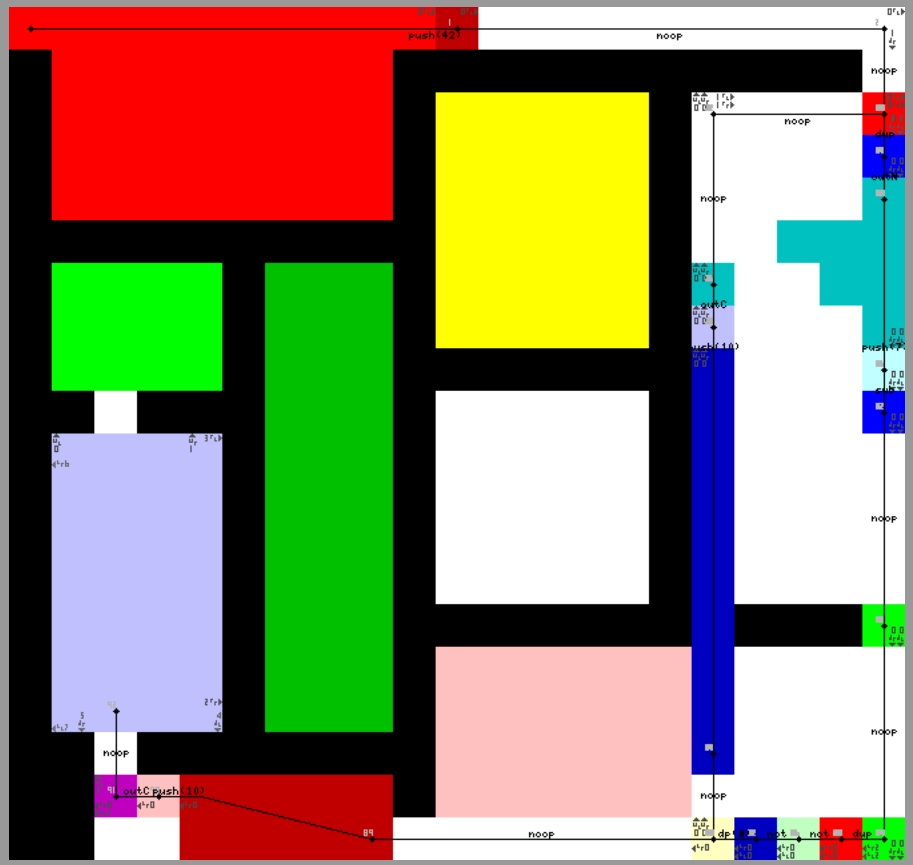


Stack

command:
push(7)

output:
42

Execution Trace



7
42

Stack

command:
sub()

output:
42

Figure: Trace of a Piet program.

Execution Trace

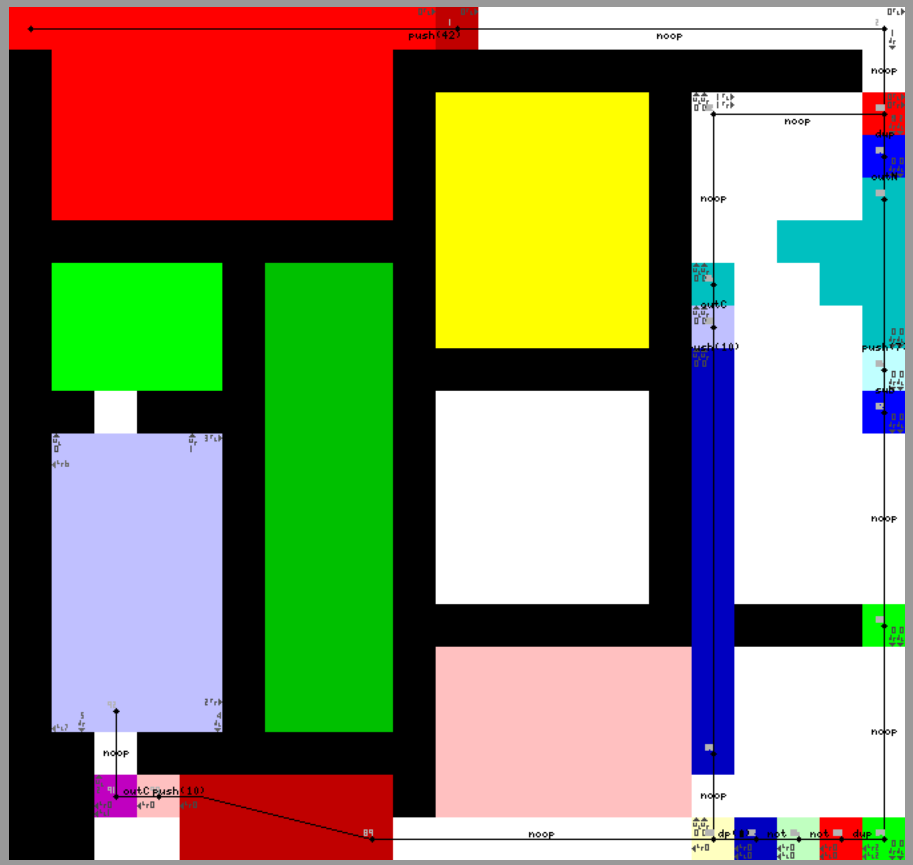
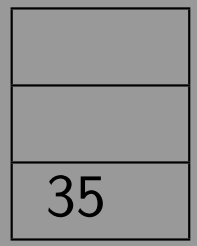


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42

Execution Trace

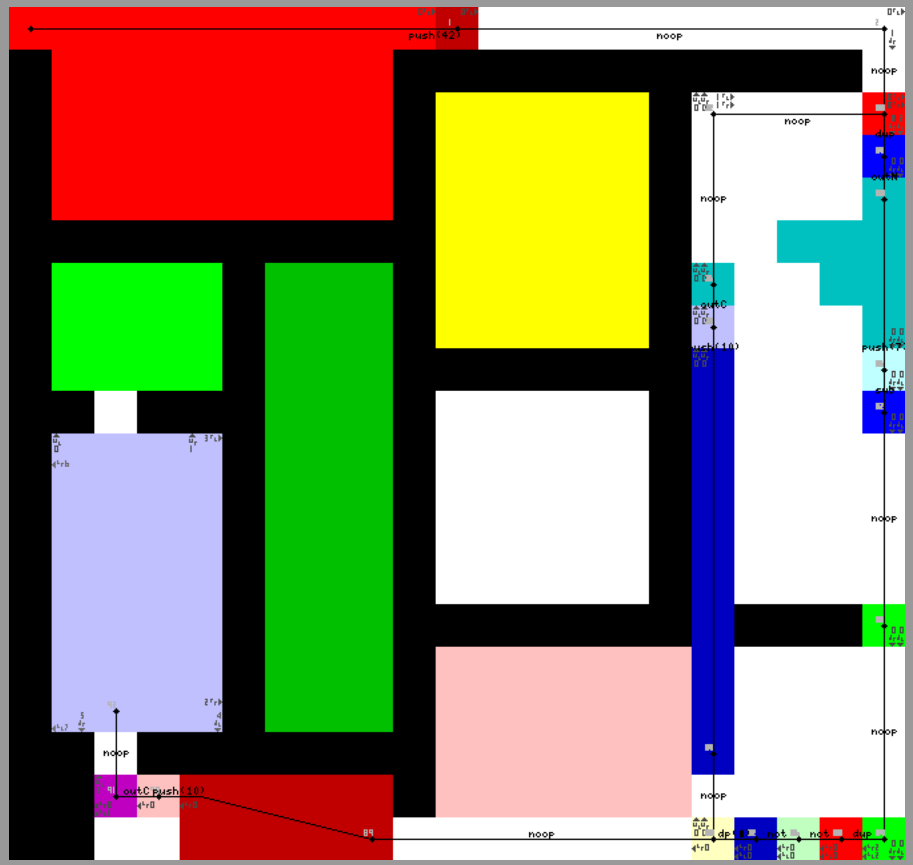


Figure: Trace of a Piet program.

35
35

Stack

command:
not()

output:
42

Execution Trace

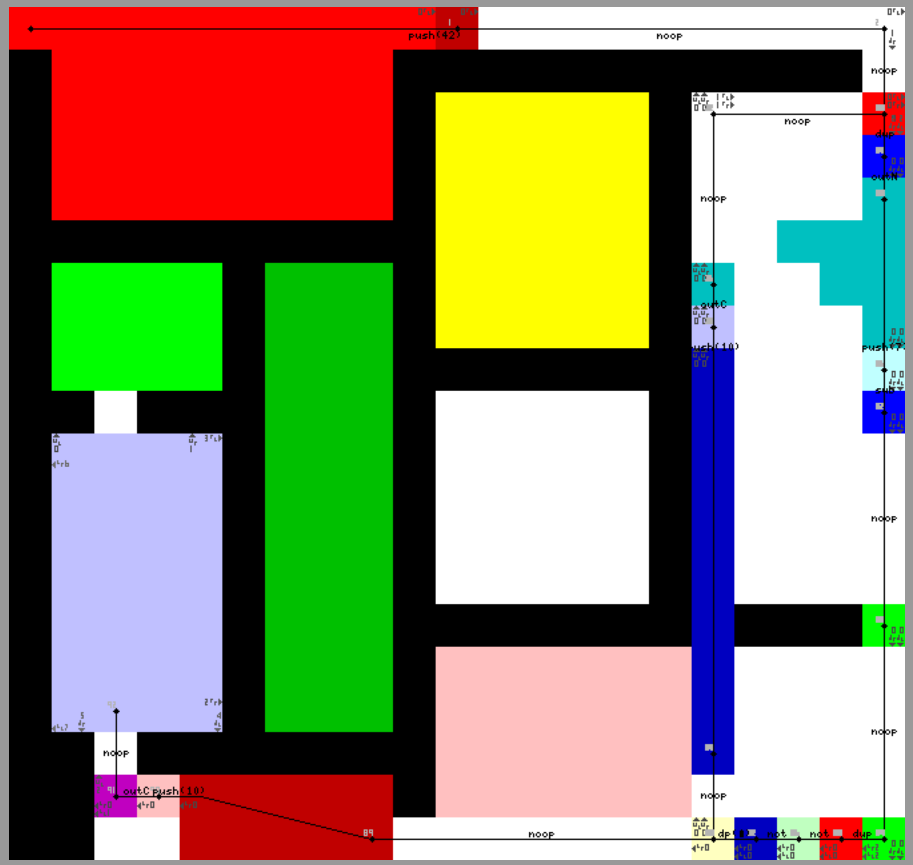


Figure: Trace of a Piet program.

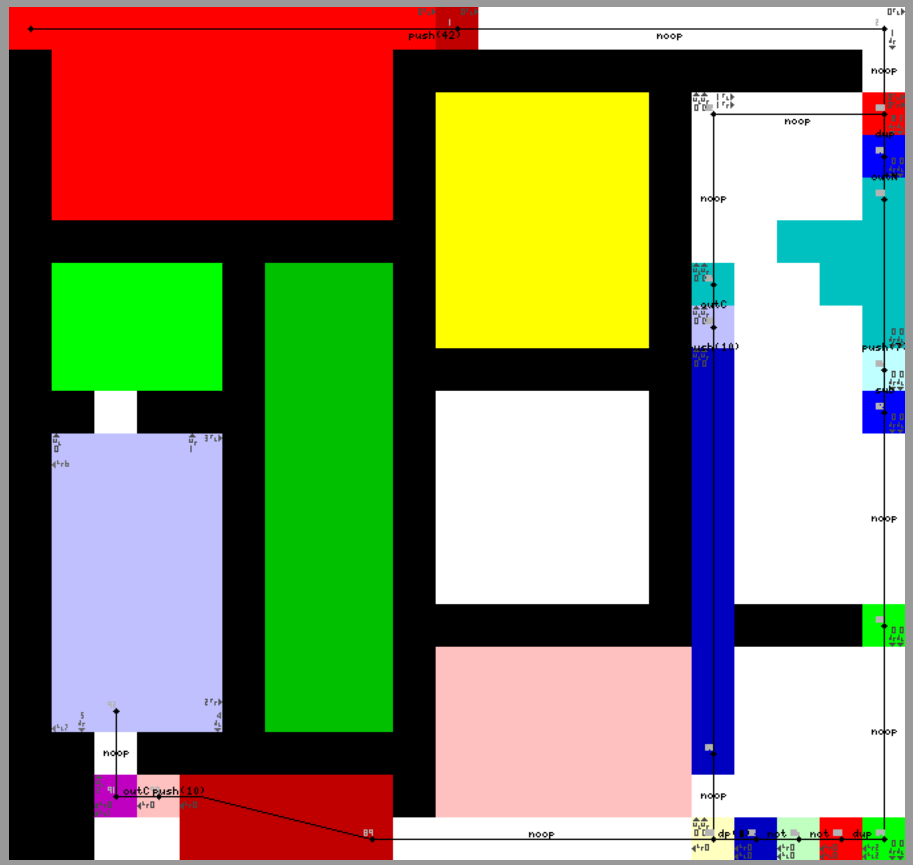
0
35

Stack

command:
not()

output:
42

Execution Trace



1
35

Stack

command:
pointer()

output:
42

Figure: Trace of a Piet program.

Execution Trace

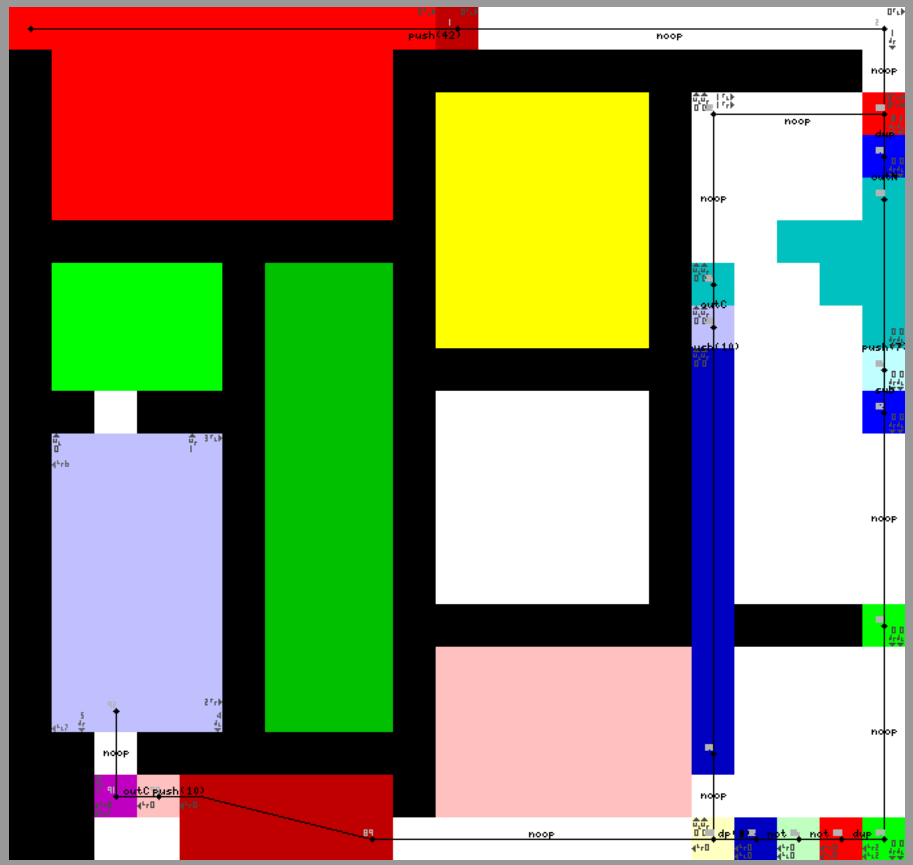
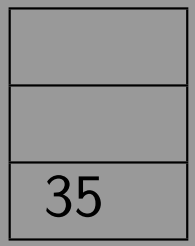


Figure: Trace of a Piet program.



Stack

command:
push(10)

output:
42

Execution Trace

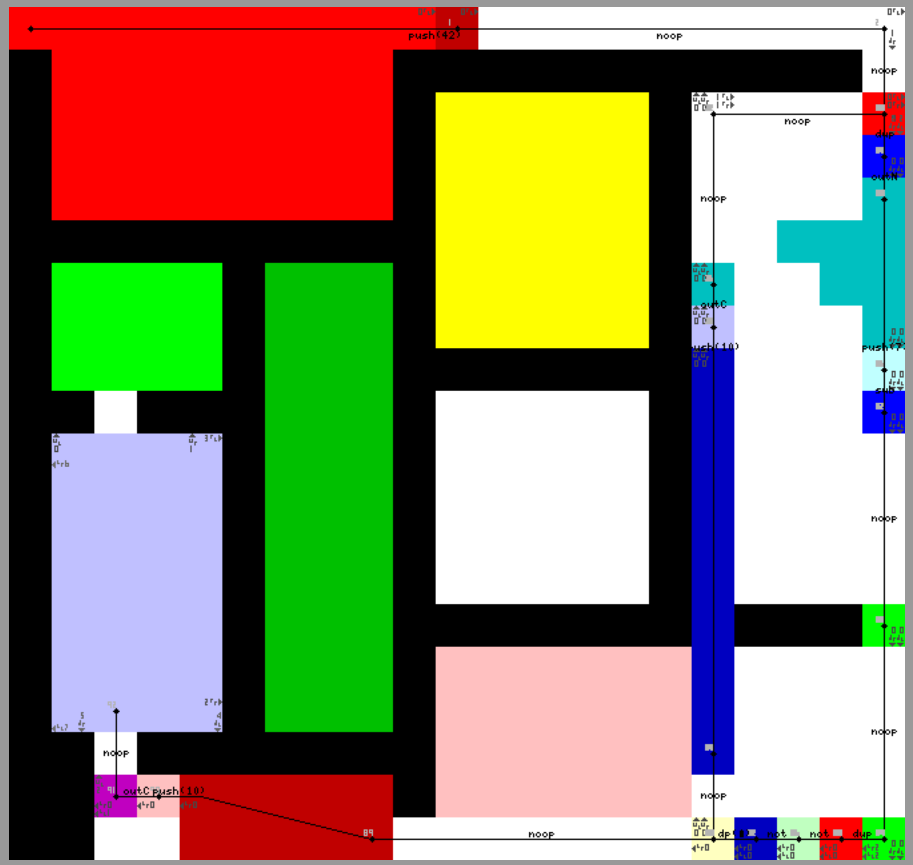


Figure: Trace of a Piet program.

10
35

Stack

command:
outC()

output:
42

Execution Trace

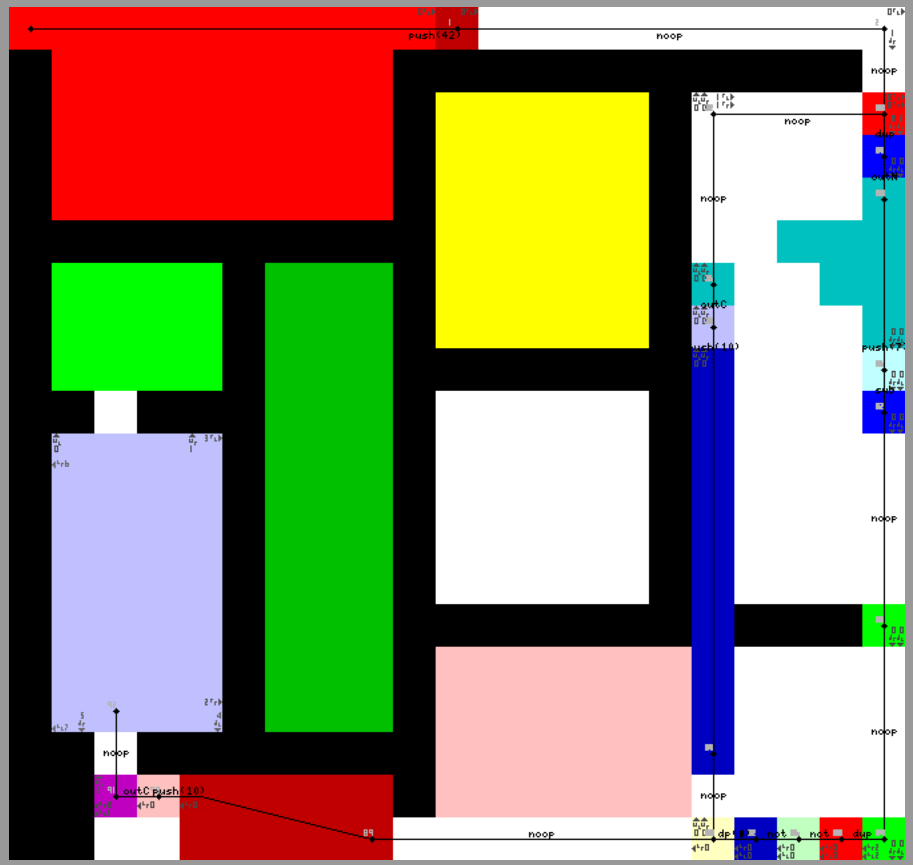
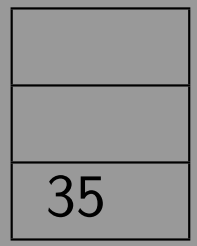


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42

Execution Trace

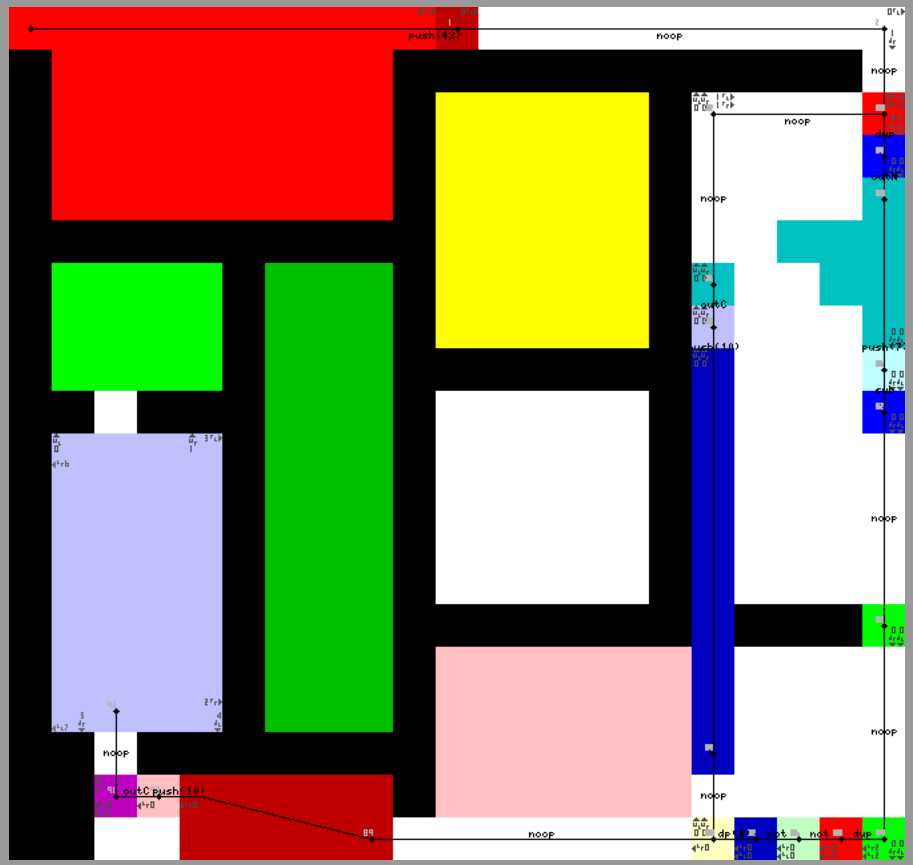


Figure: Trace of a Piet program.

35
35

Stack

command:
outN()

output:
42

Execution Trace

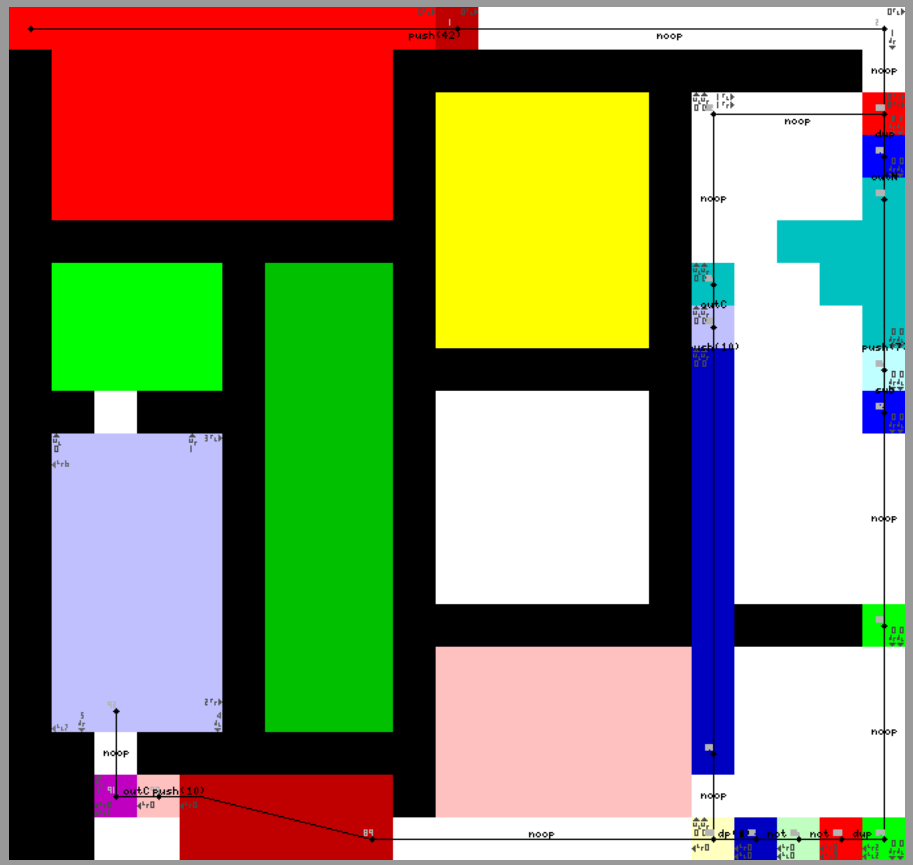
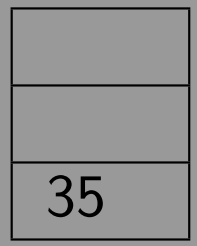


Figure: Trace of a Piet program.



Stack

command:

output:

42
35

Execution Trace

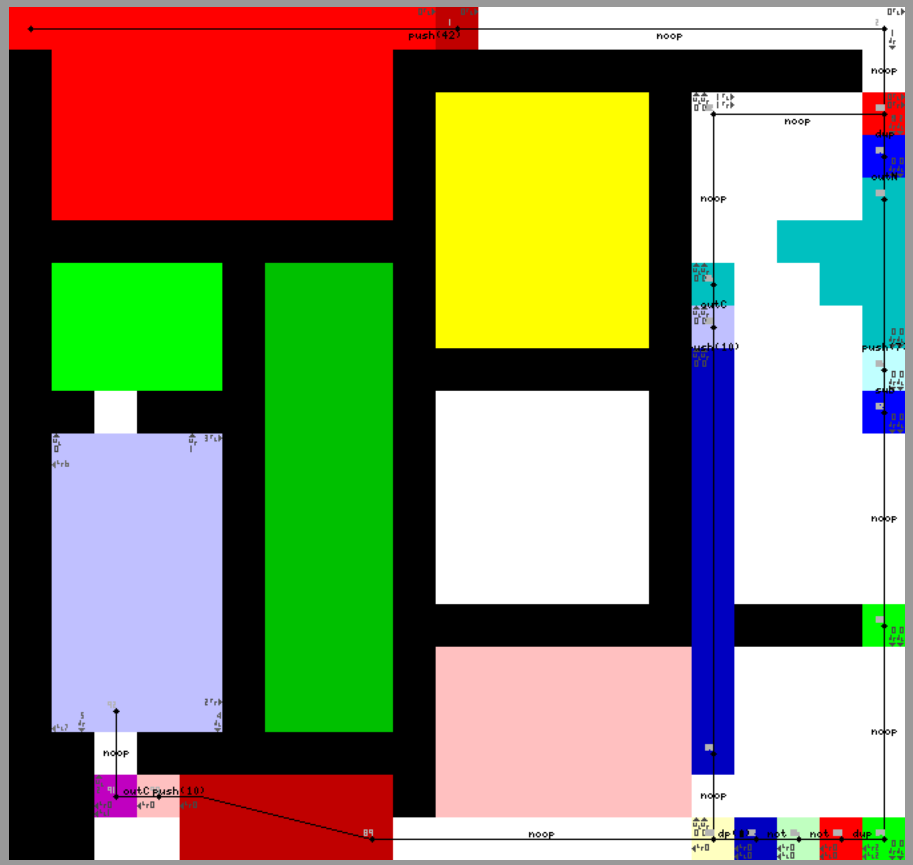
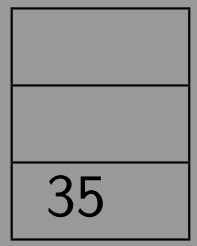


Figure: Trace of a Piet program.



Stack

command:
push(7)

output:
42
35

Execution Trace

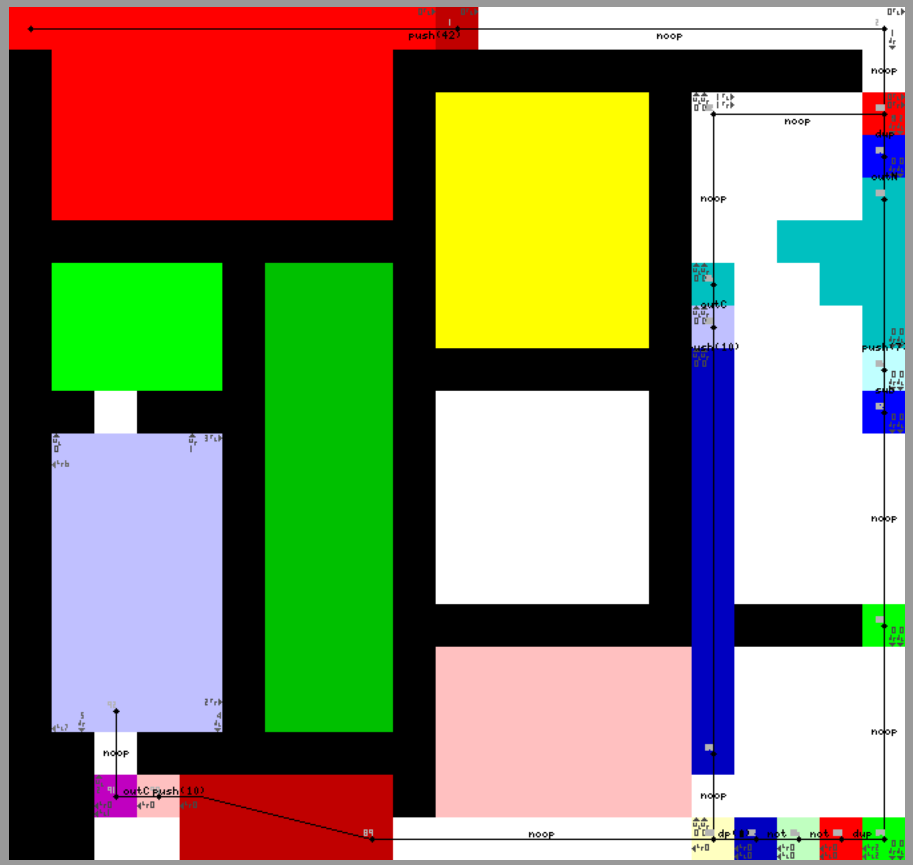


Figure: Trace of a Piet program.

7
35

Stack

command:
sub()

output:
42
35

Execution Trace

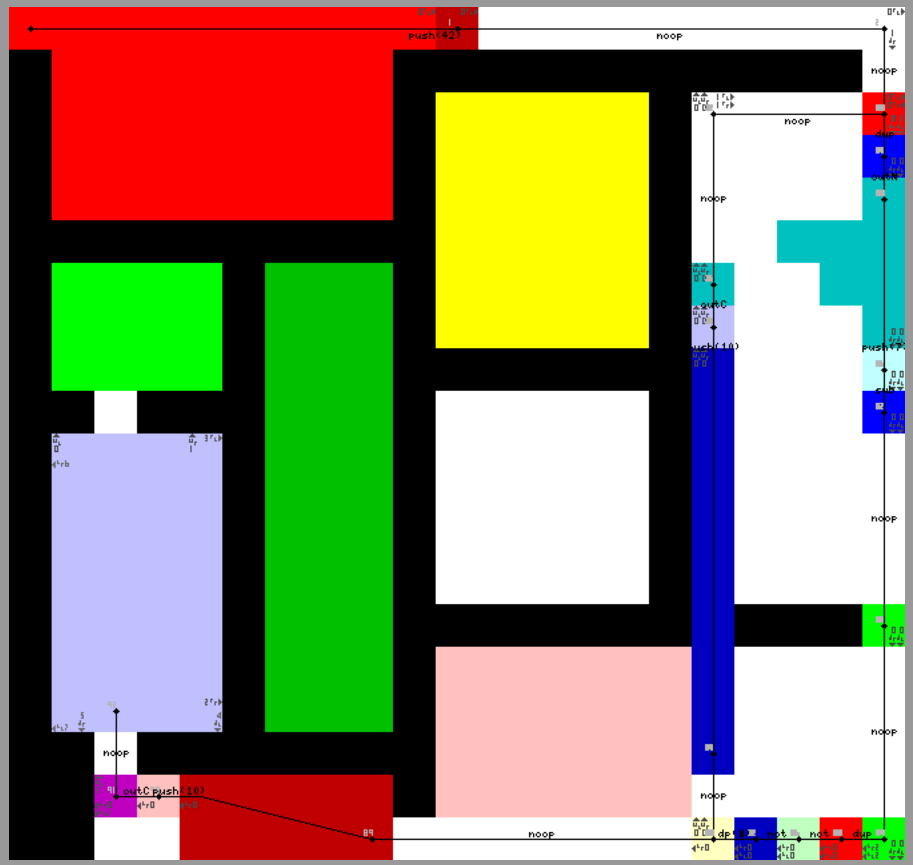
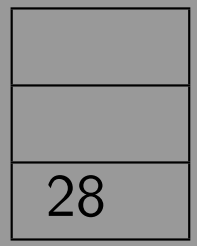


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35

Execution Trace

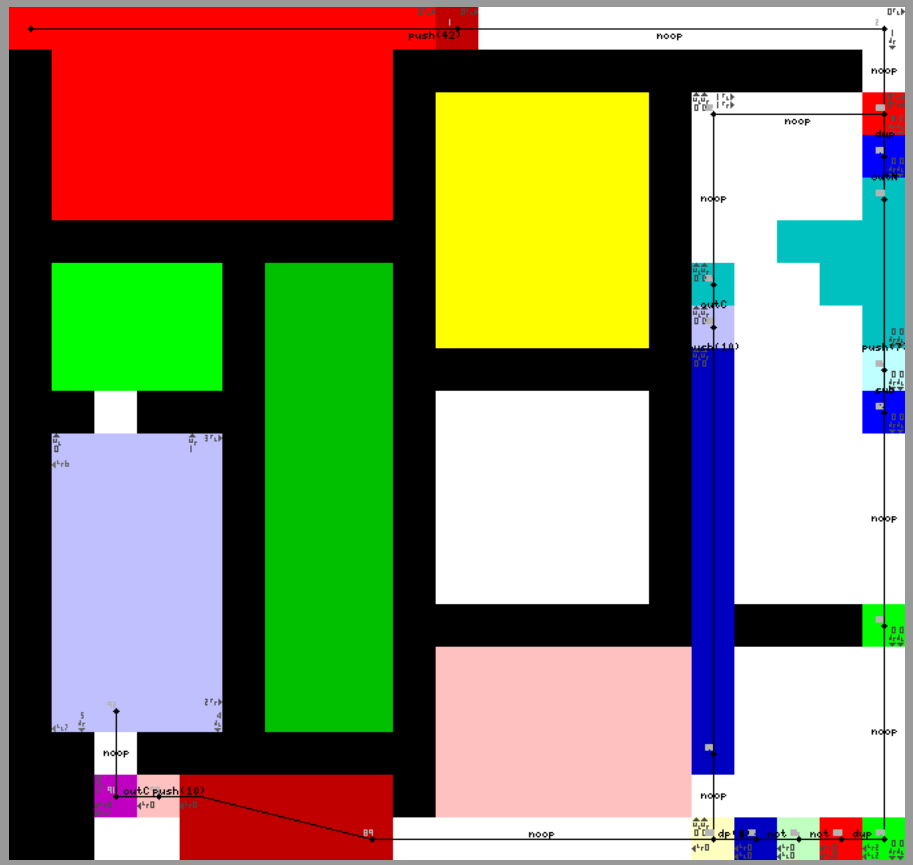


Figure: Trace of a Piet program.

28
28

Stack

command:
not()

output:
42
35

Execution Trace

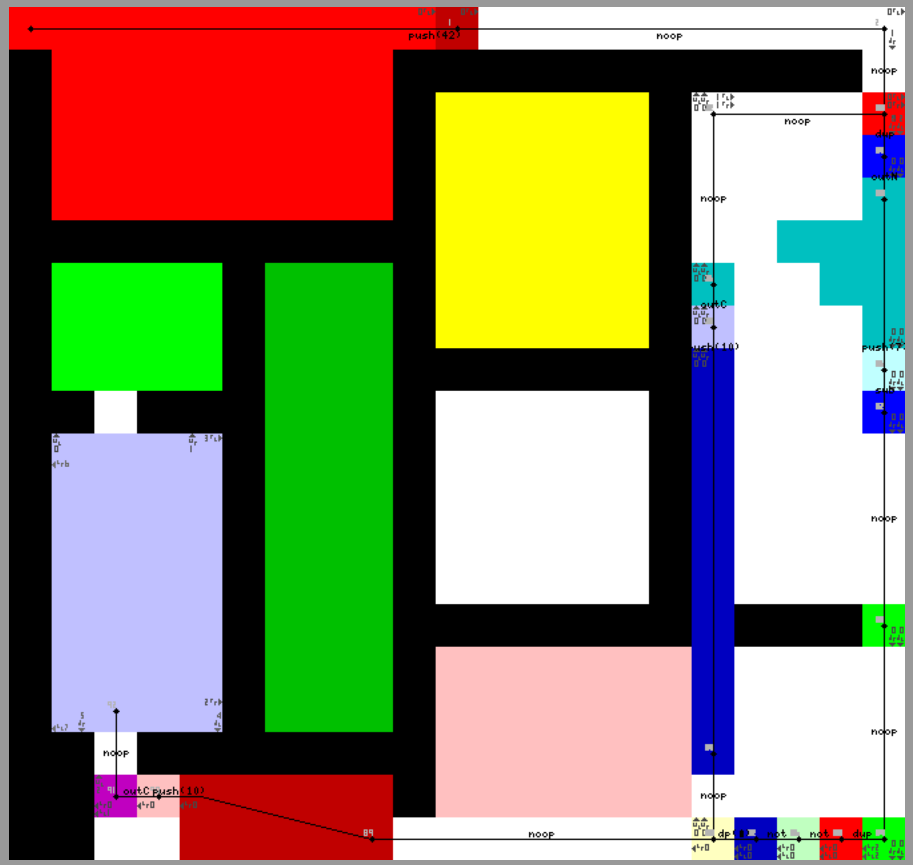


Figure: Trace of a Piet program.

0
28

Stack

command:
not()

output:
42
35

Execution Trace

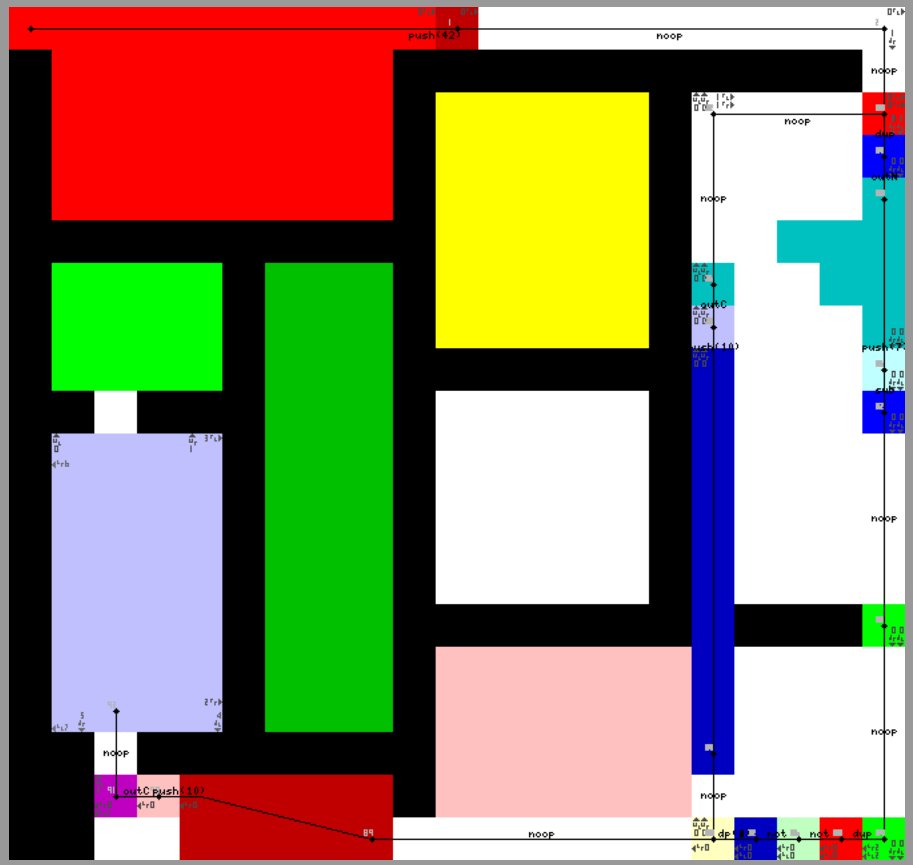


Figure: Trace of a Piet program.

1
28

Stack

command:
pointer()

output:
42
35

Execution Trace

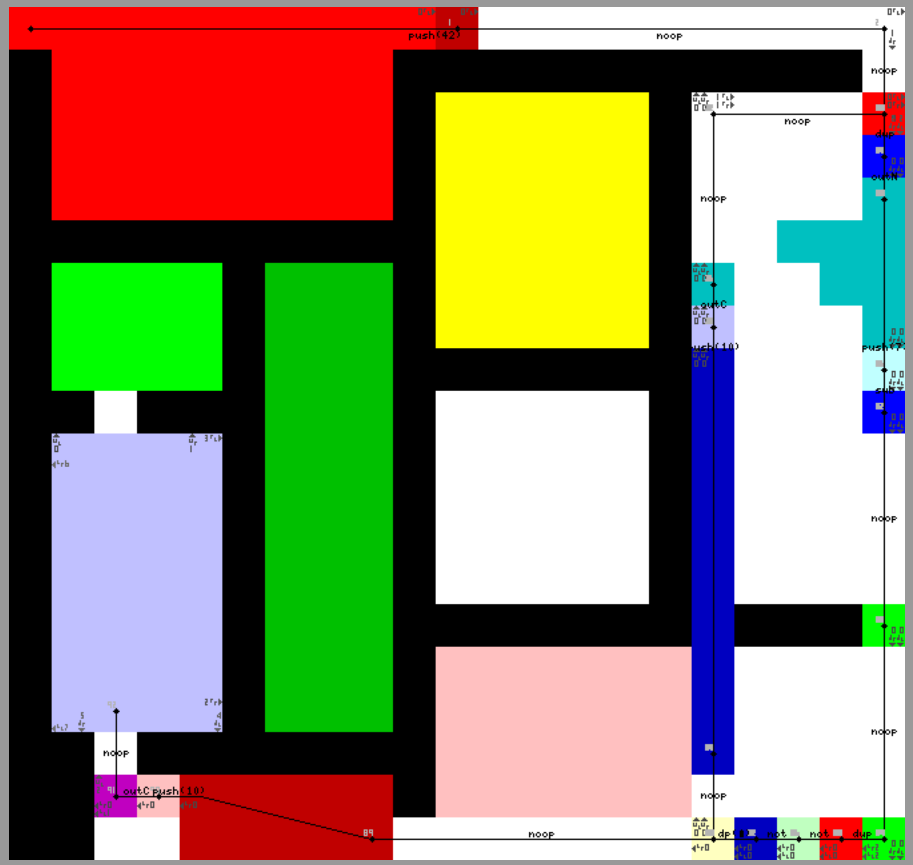
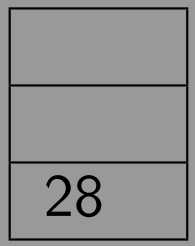


Figure: Trace of a Piet program.



Stack

command:
push(10)

output:
42
35

Execution Trace

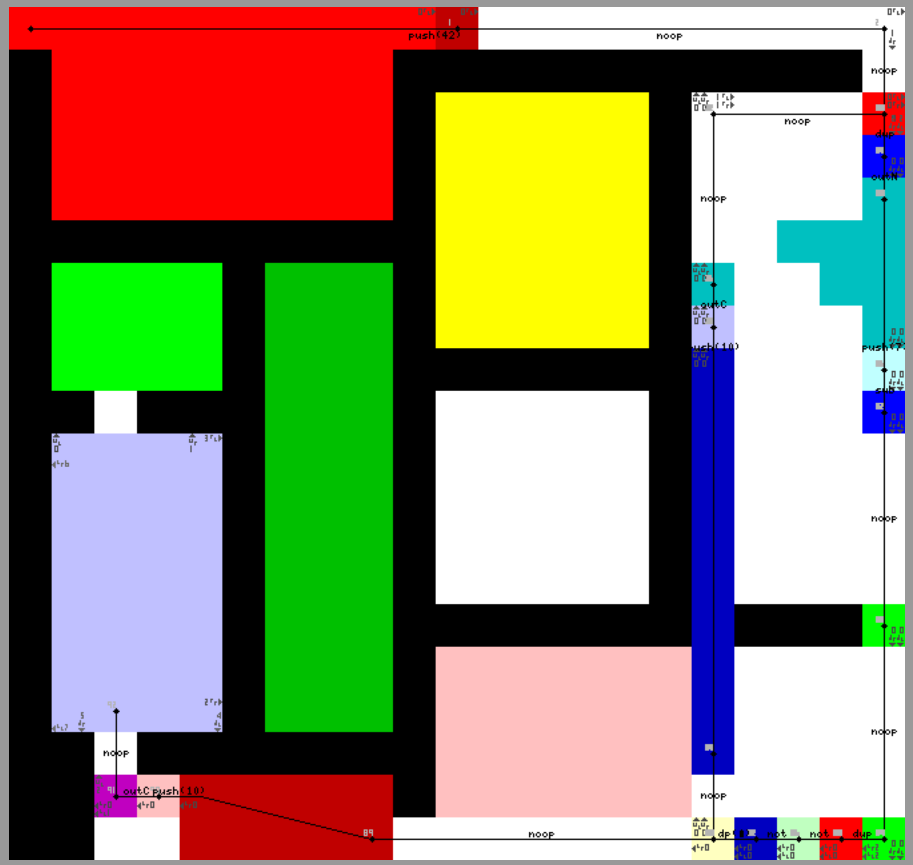


Figure: Trace of a Piet program.

10
28

Stack

command:
outC()

output:
42
35

Execution Trace

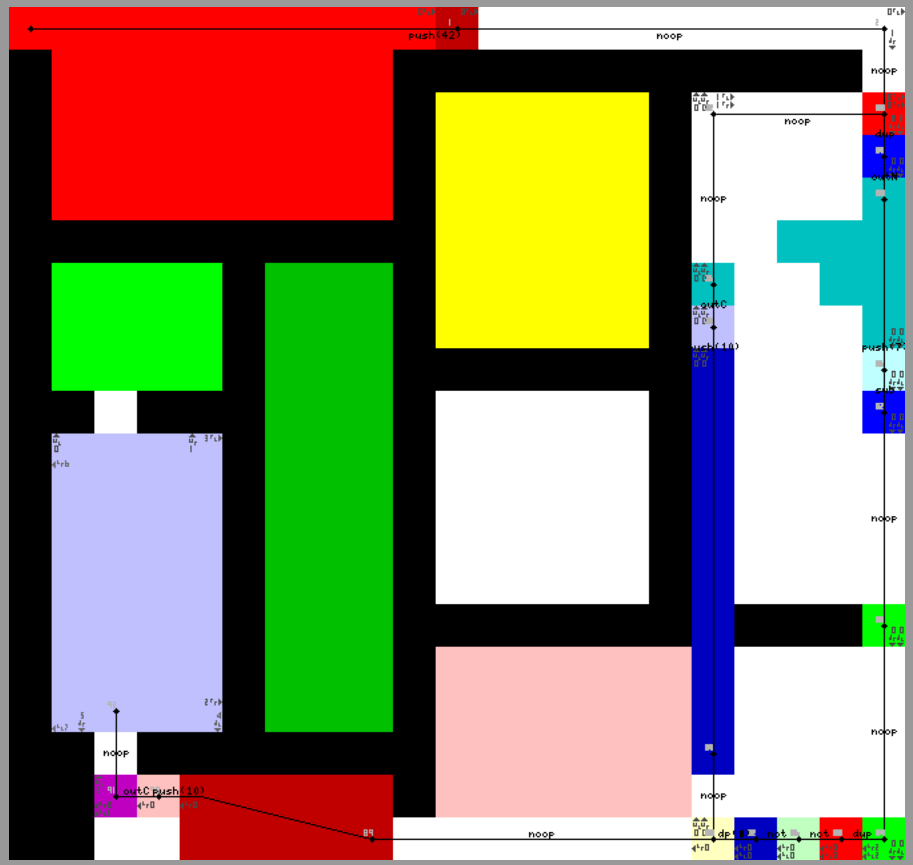
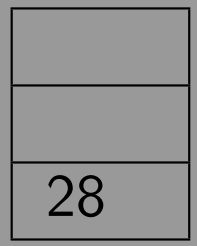


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35

Execution Trace

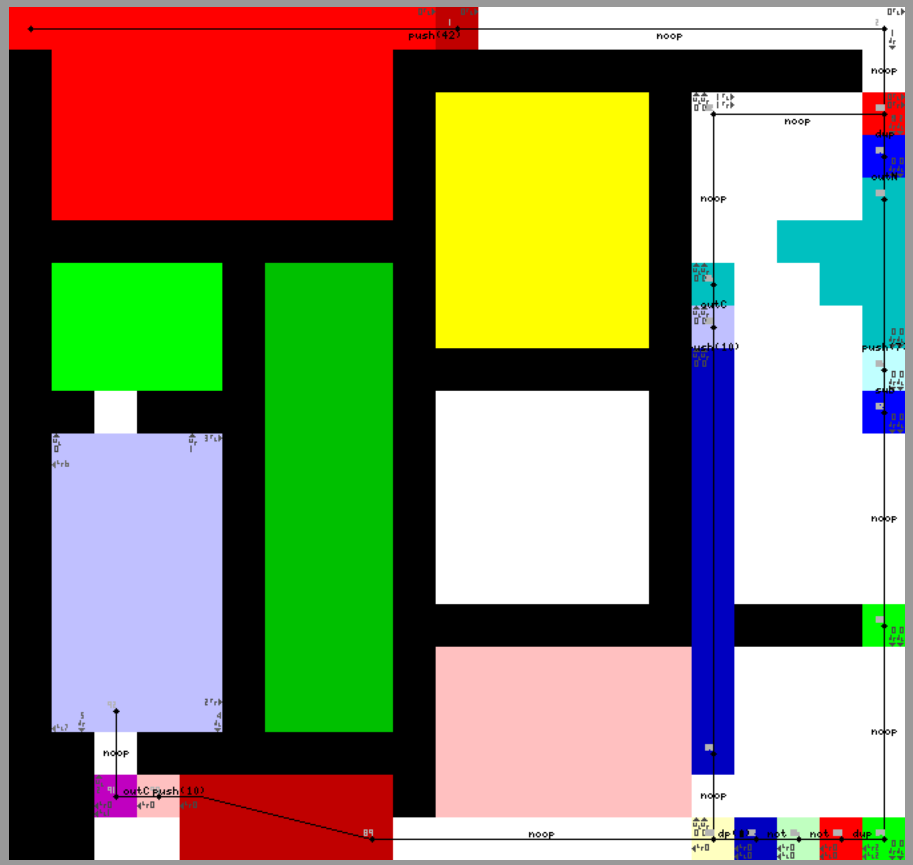


Figure: Trace of a Piet program.

28
28

Stack

command:
outN()

output:
42
35

Execution Trace

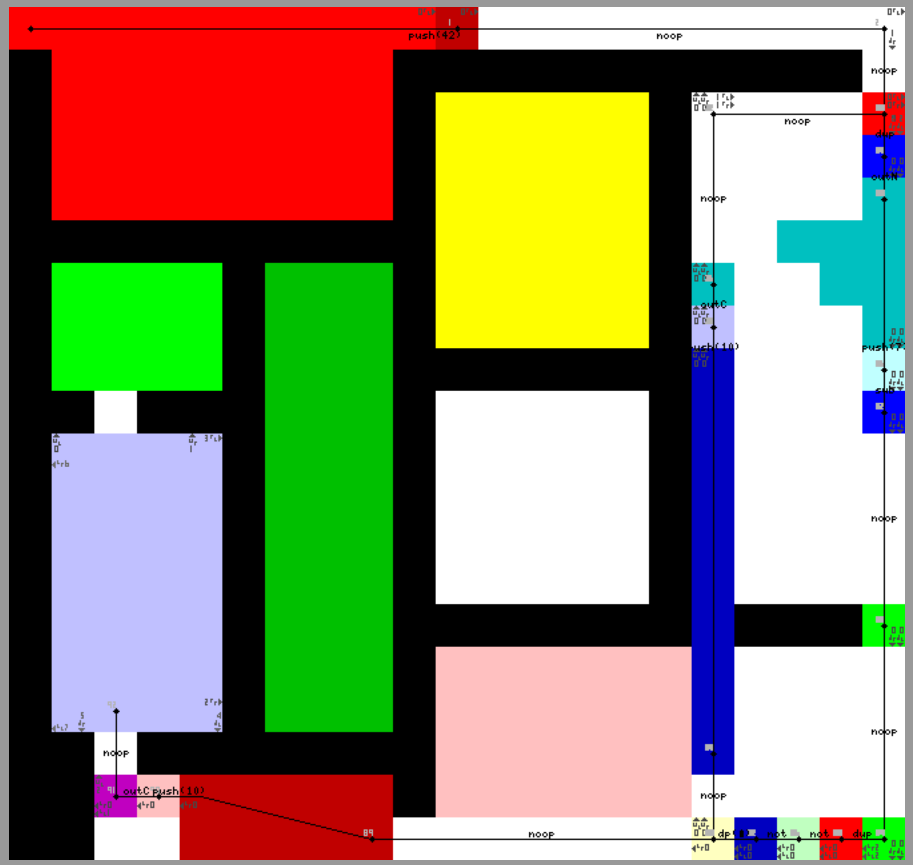
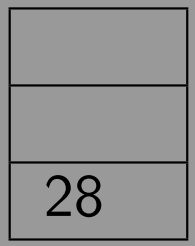


Figure: Trace of a Piet program.



Stack

command:

output:

42
35
28

Execution Trace

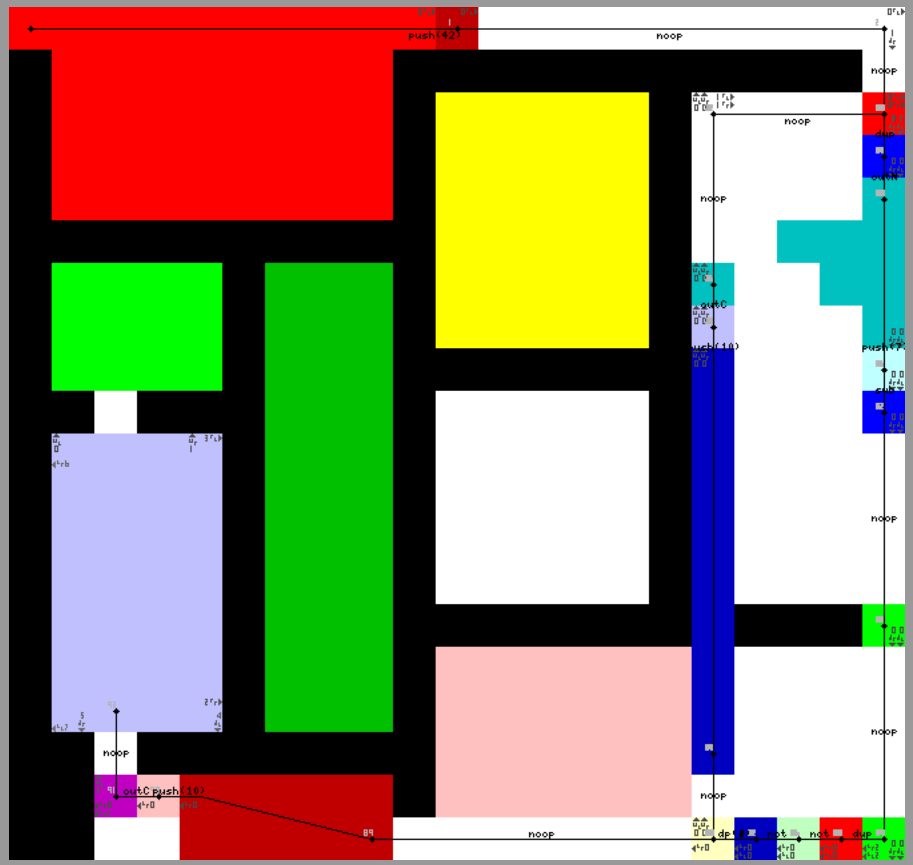
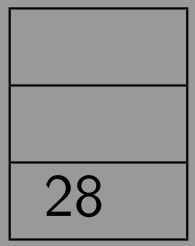


Figure: Trace of a Piet program.



Stack

command:
push(7)

output:
42
35
28

Execution Trace

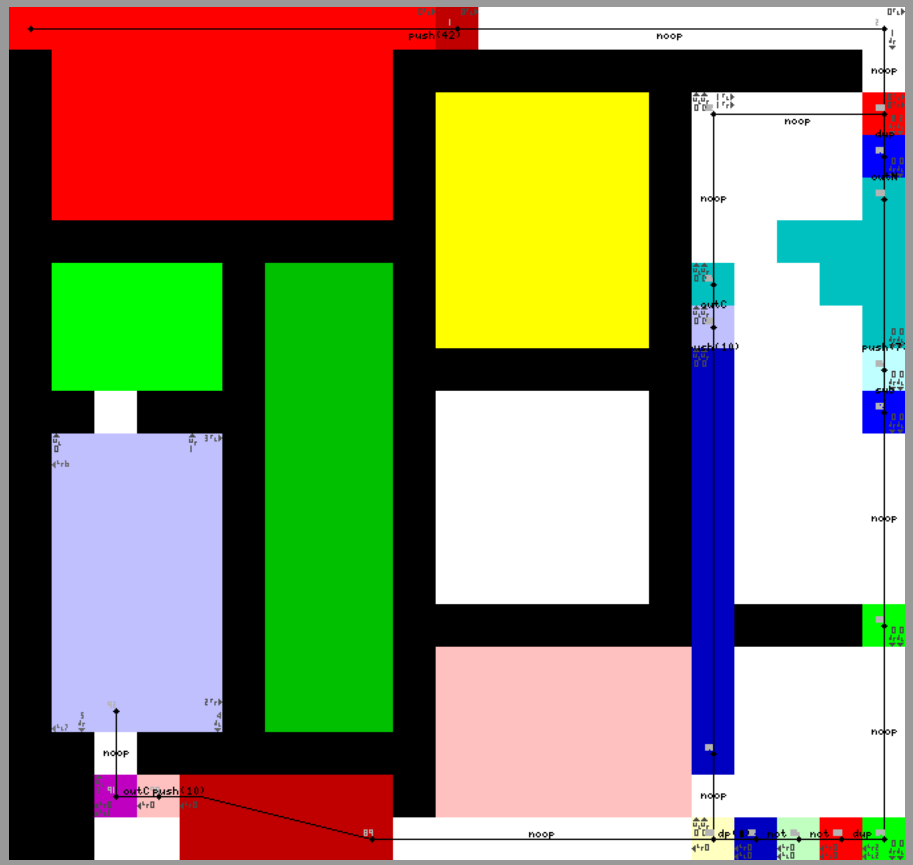


Figure: Trace of a Piet program.

7
28

Stack

command:
sub()

output:
42
35
28

Execution Trace

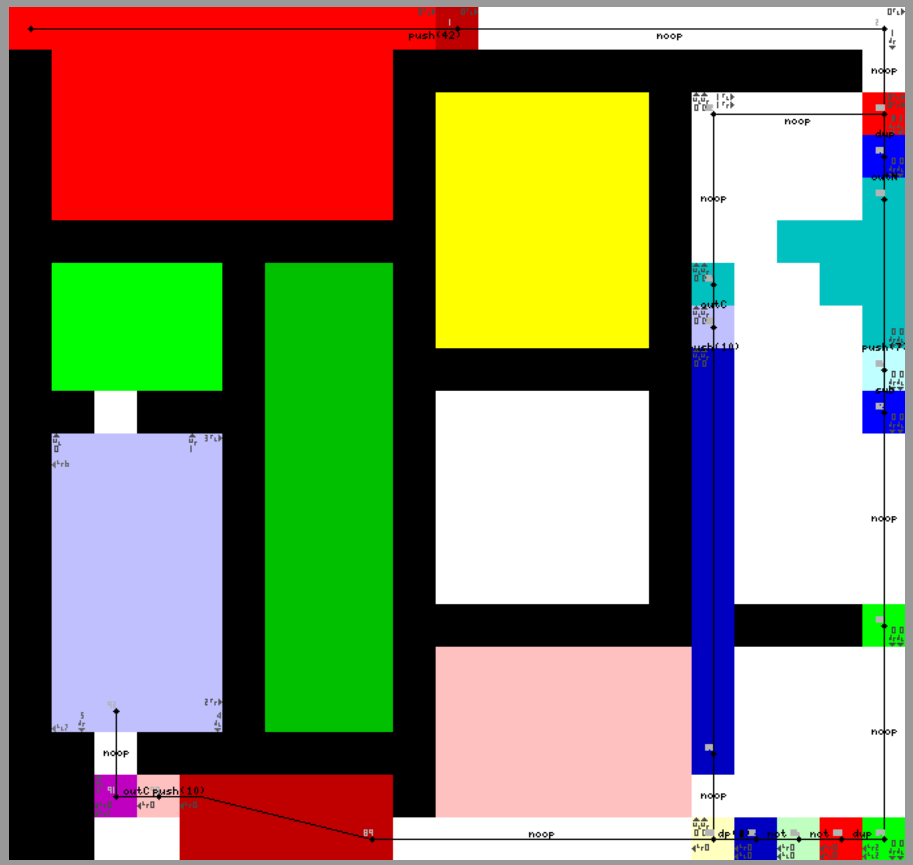
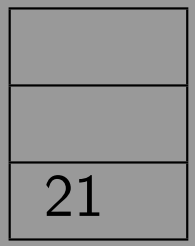


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35
28

Execution Trace

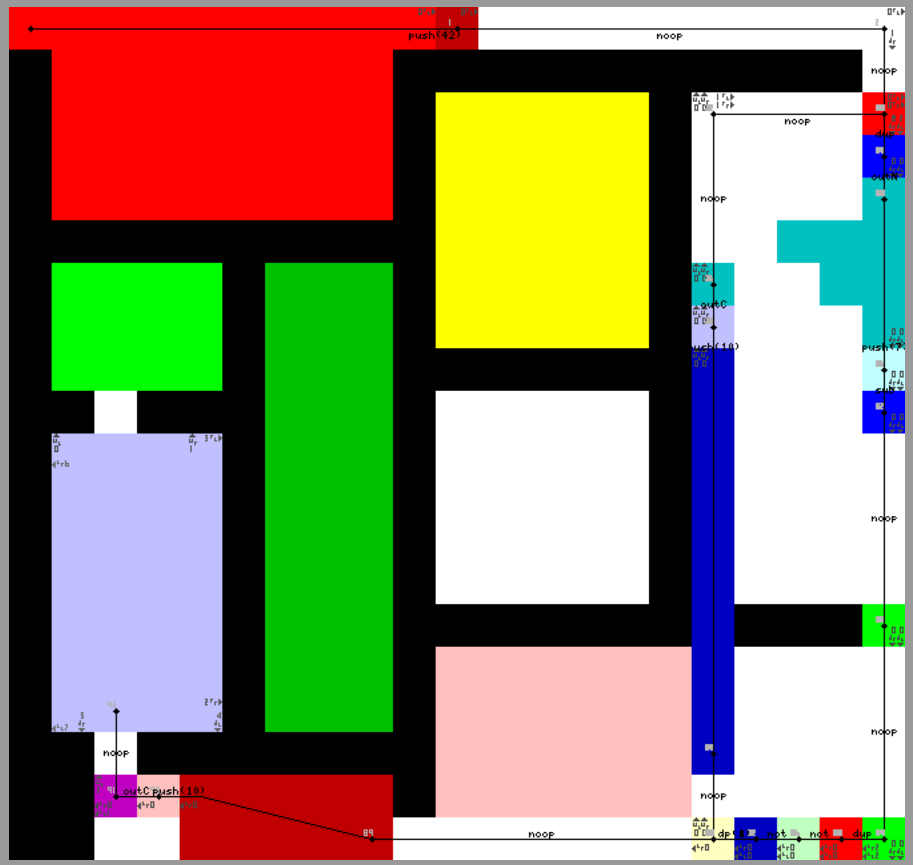


Figure: Trace of a Piet program.

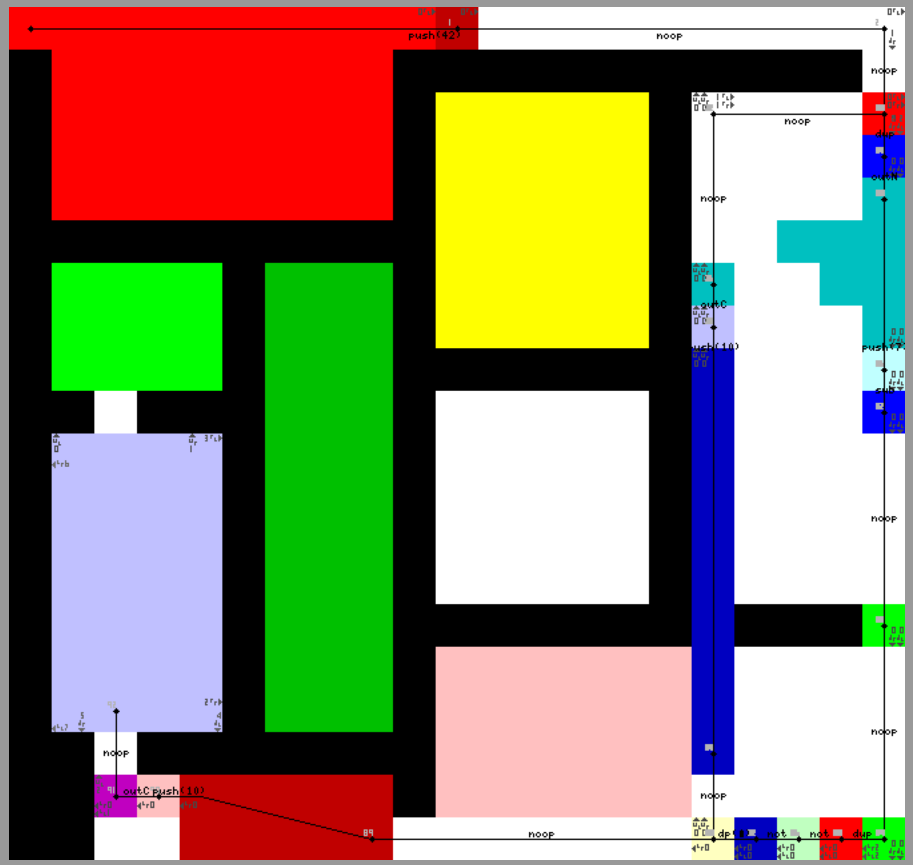
21
21

Stack

command:
not()

output:
42
35
28

Execution Trace



0
21

Stack

command:
not()

output:
42
35
28

Figure: Trace of a Piet program.

Execution Trace

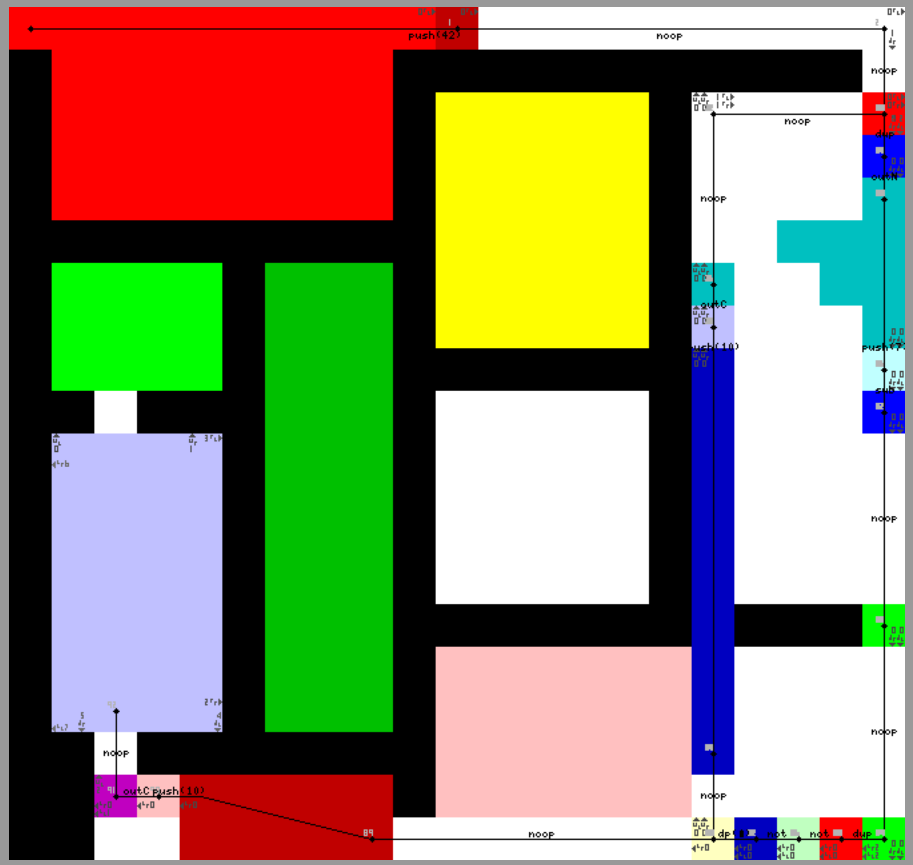


Figure: Trace of a Piet program.

1
21

Stack

command:
pointer()

output:
42
35
28

Execution Trace

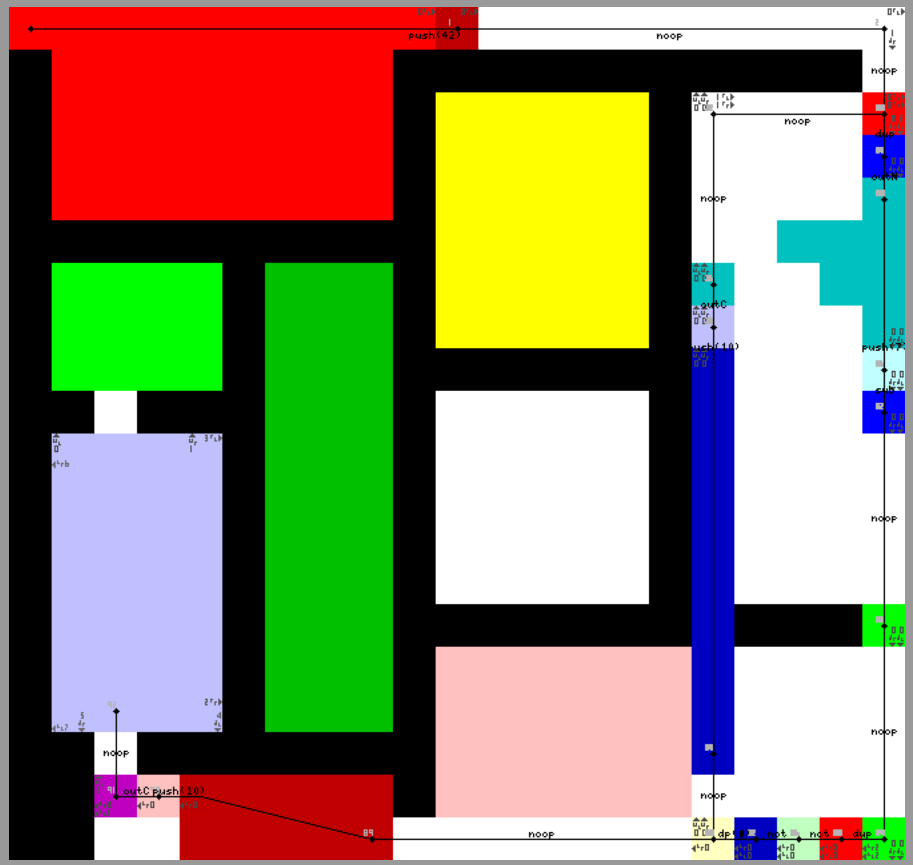
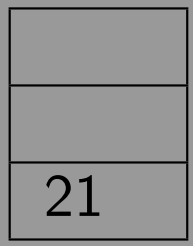


Figure: Trace of a Piet program.



Stack

command:
push(10)

output:
42
35
28

Execution Trace

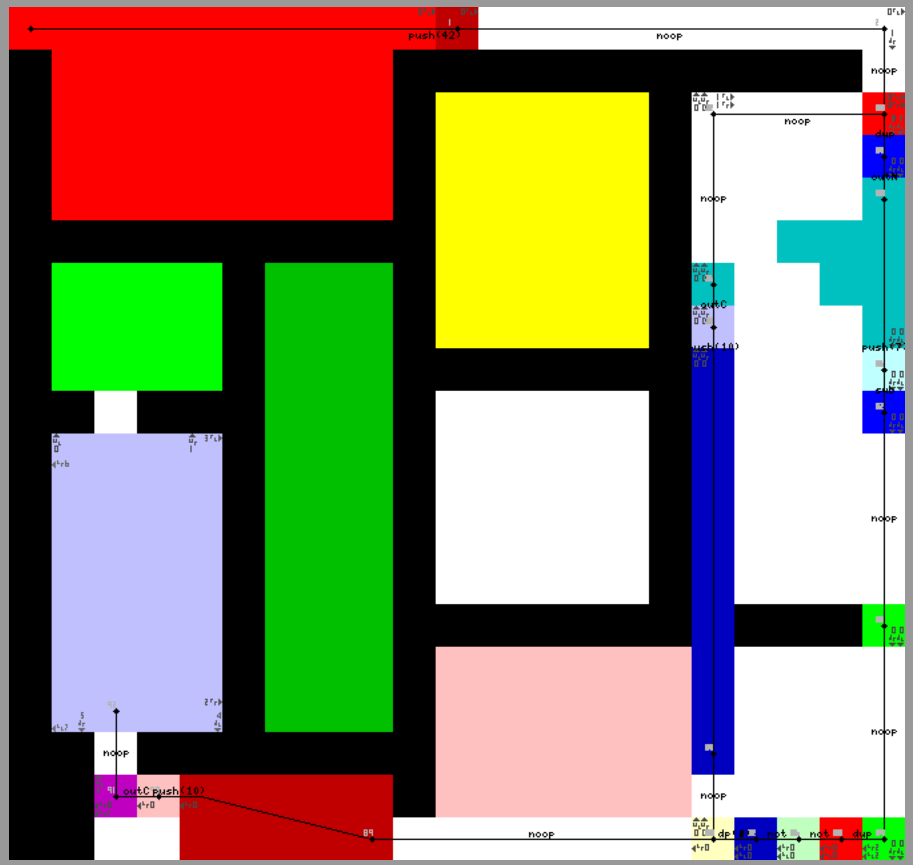


Figure: Trace of a Piet program.

10
21

Stack

command:
outC()

output:
42
35
28

Execution Trace

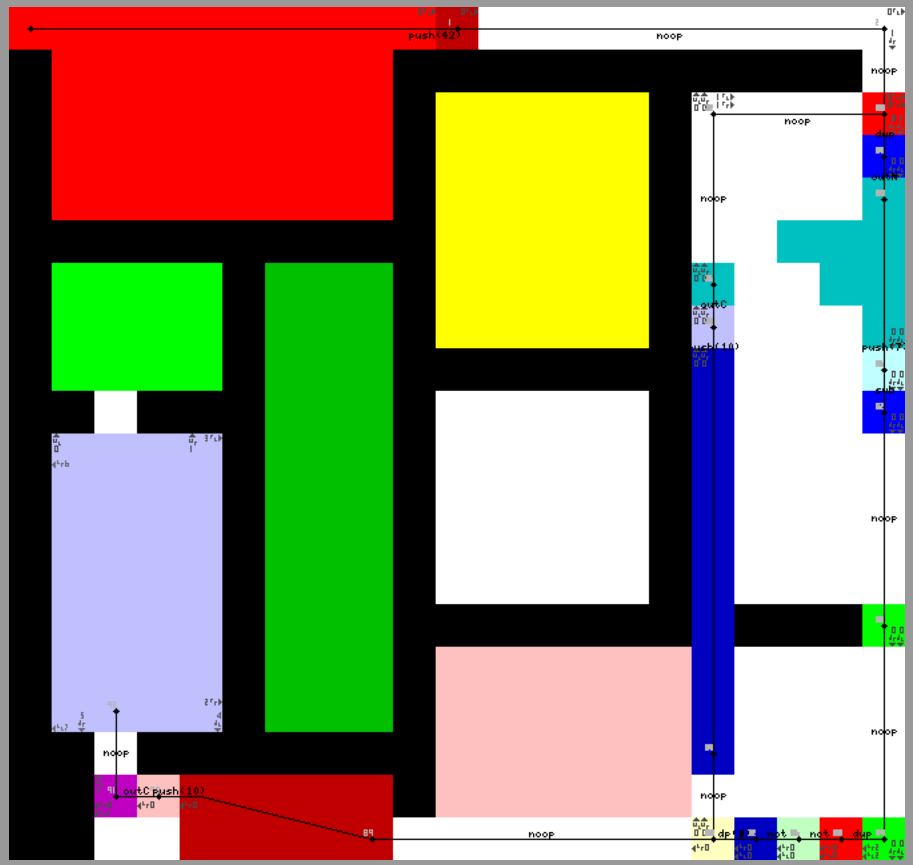
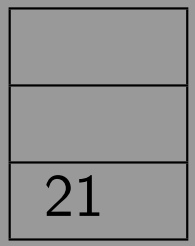


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35
28

Execution Trace

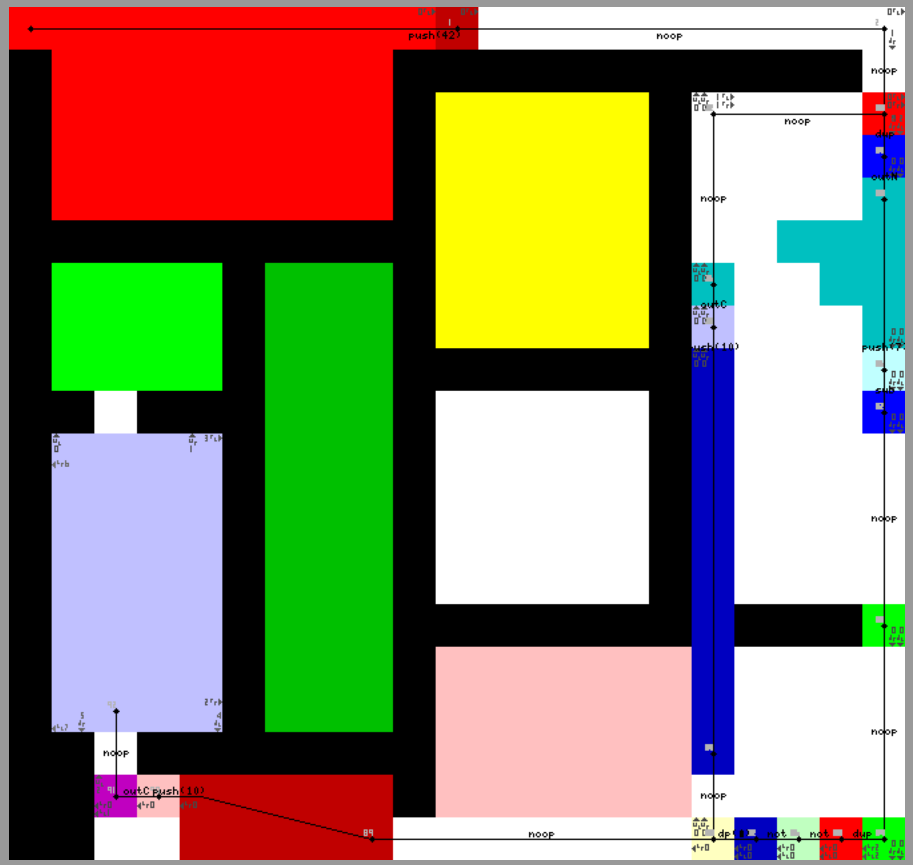


Figure: Trace of a Piet program.

21
21

Stack

command:
outN()

output:
42
35
28

Execution Trace

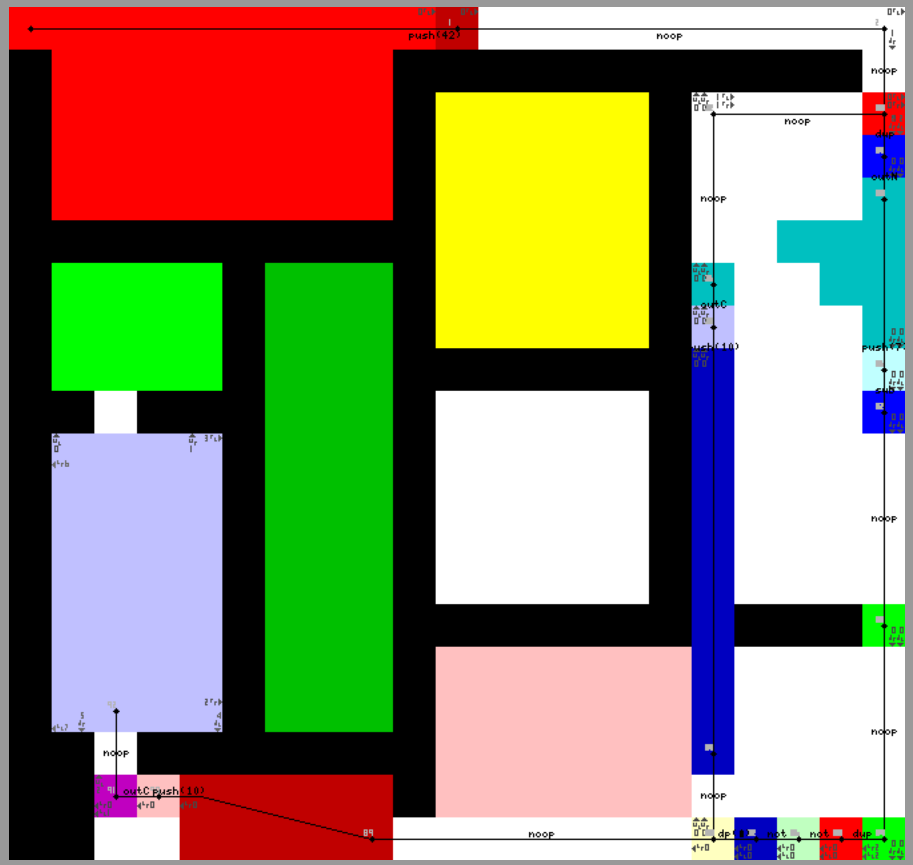
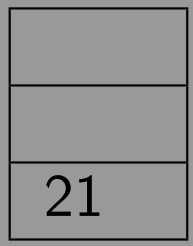


Figure: Trace of a Piet program.



Stack

command:

output:

42
35
28
21

Execution Trace

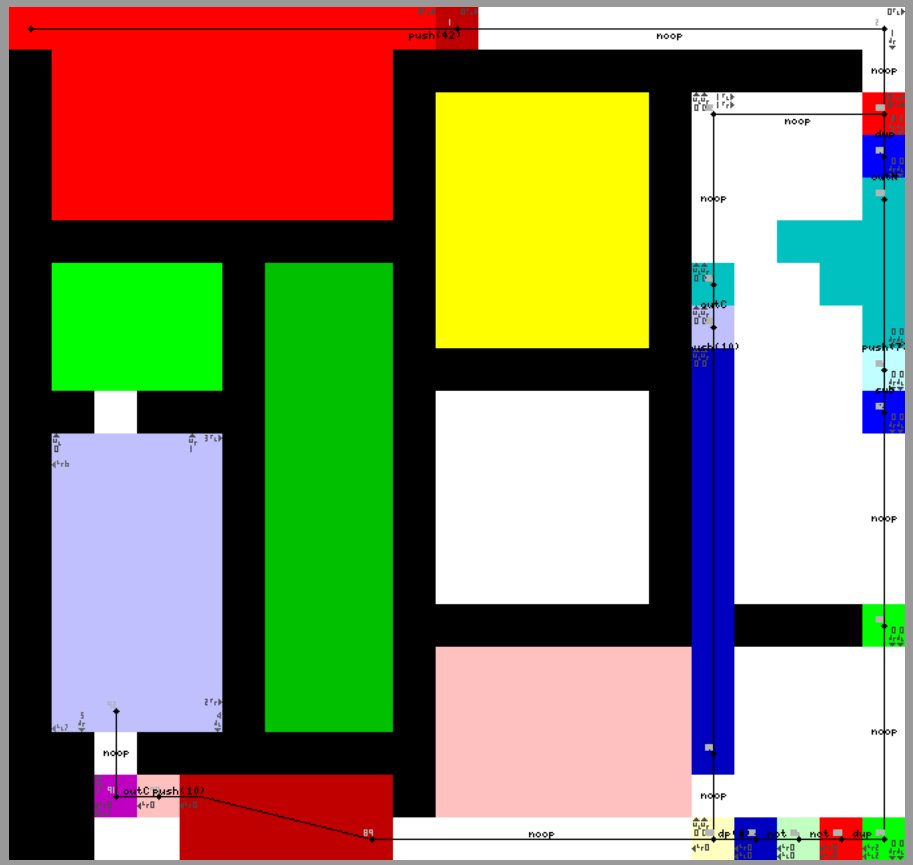
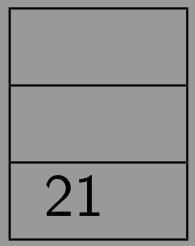


Figure: Trace of a Piet program.



Stack

command:
push(7)

output:
42
35
28
21

Execution Trace

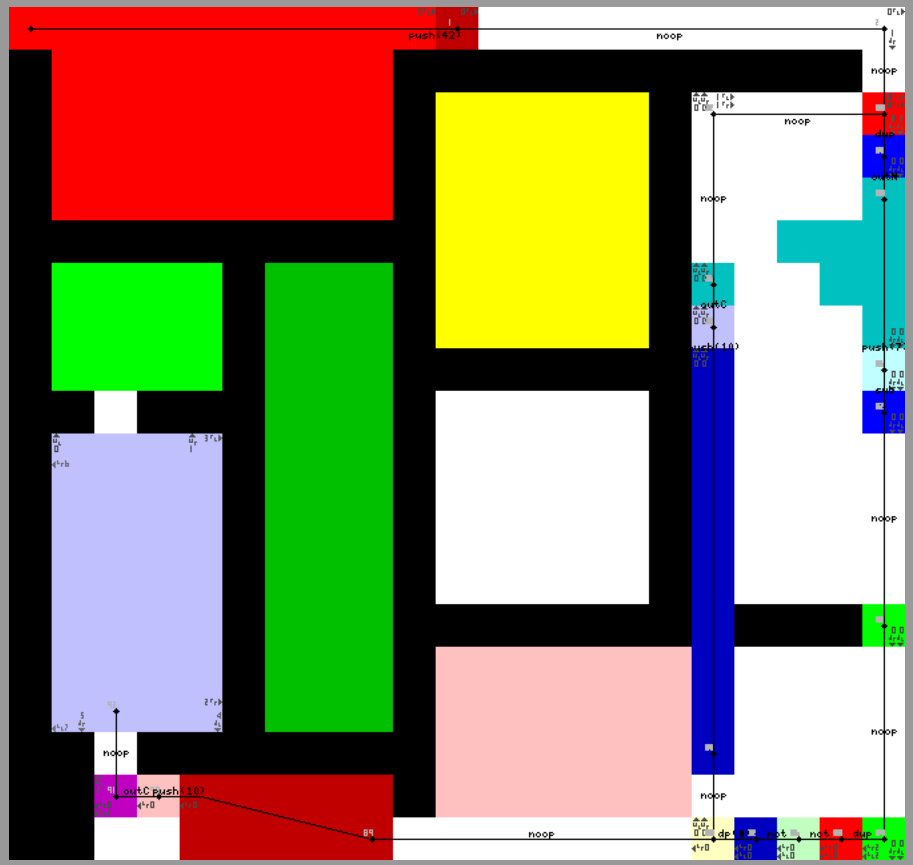


Figure: Trace of a Piet program.

7
21

Stack

command:
sub()

output:
42
35
28
21

Execution Trace

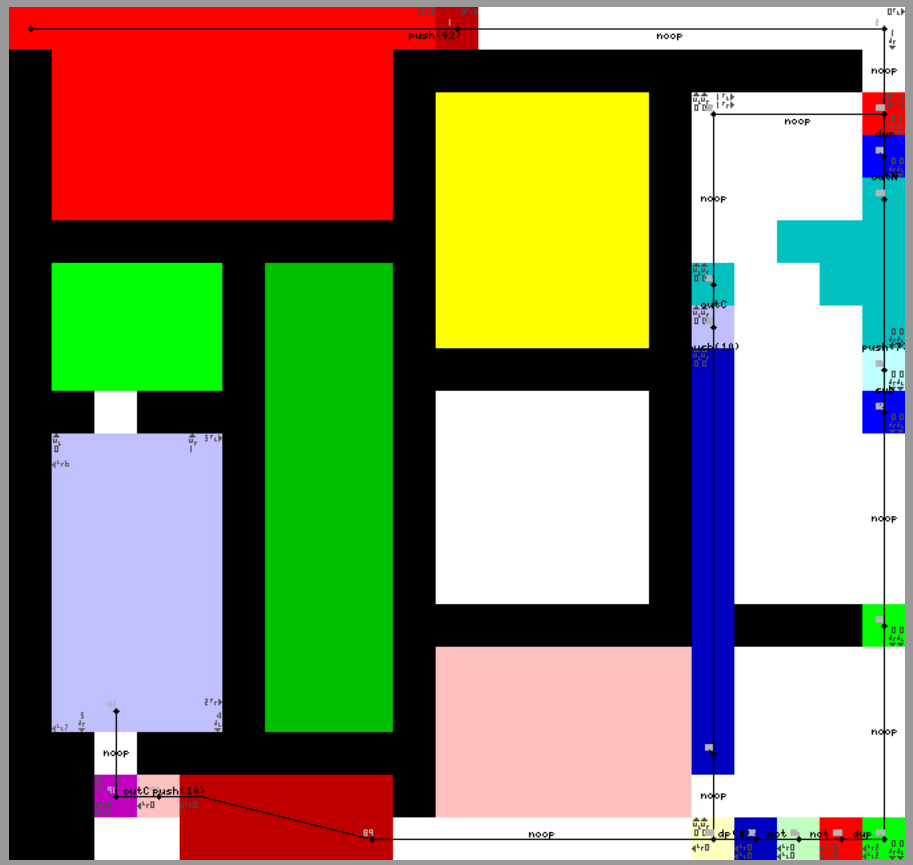
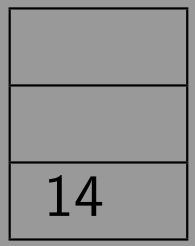


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35
28
21

Execution Trace

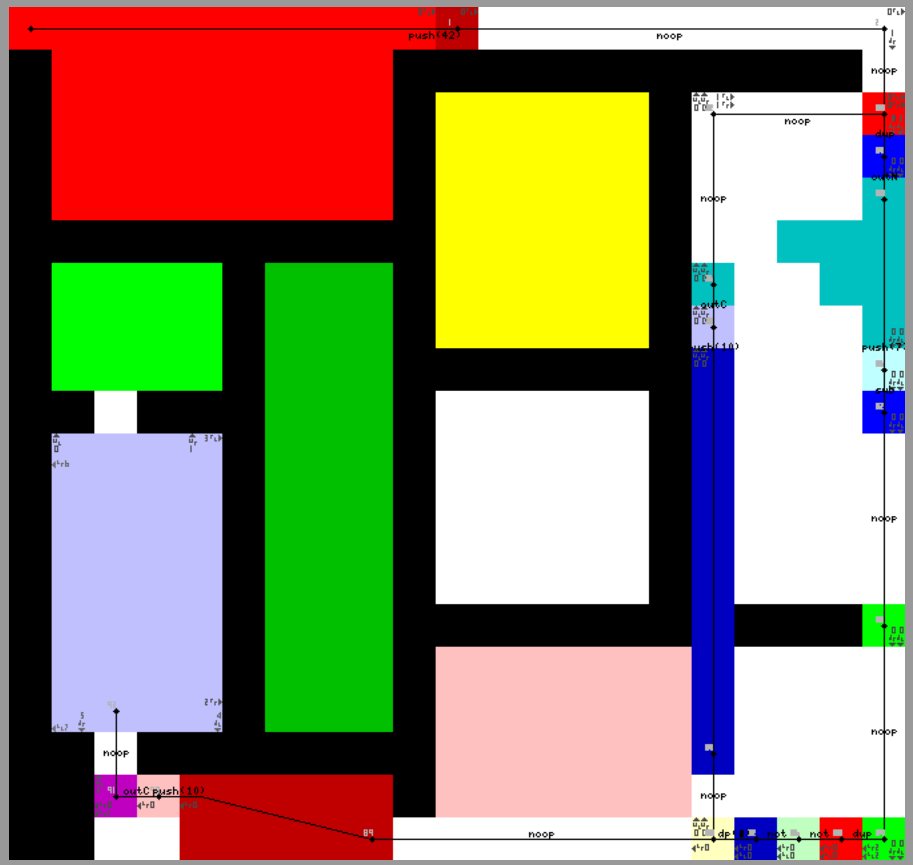


Figure: Trace of a Piet program.

14
14

Stack

command:
not()

output:
42
35
28
21

Execution Trace

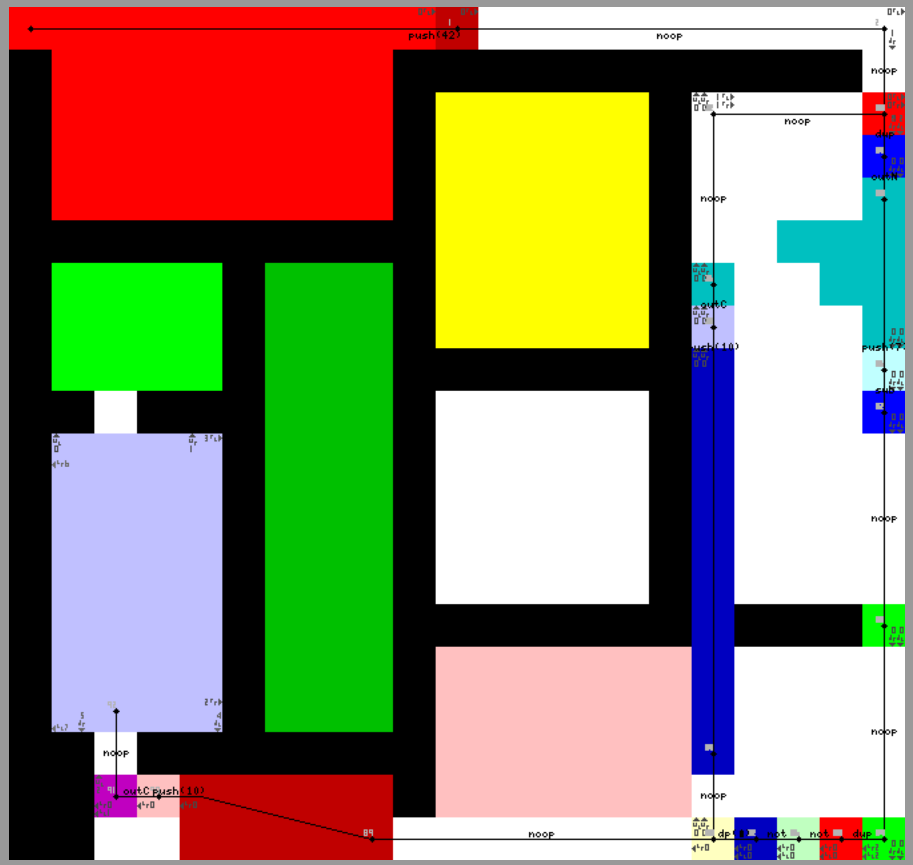


Figure: Trace of a Piet program.

0
14

Stack

command:
not()

output:
42
35
28
21

Execution Trace

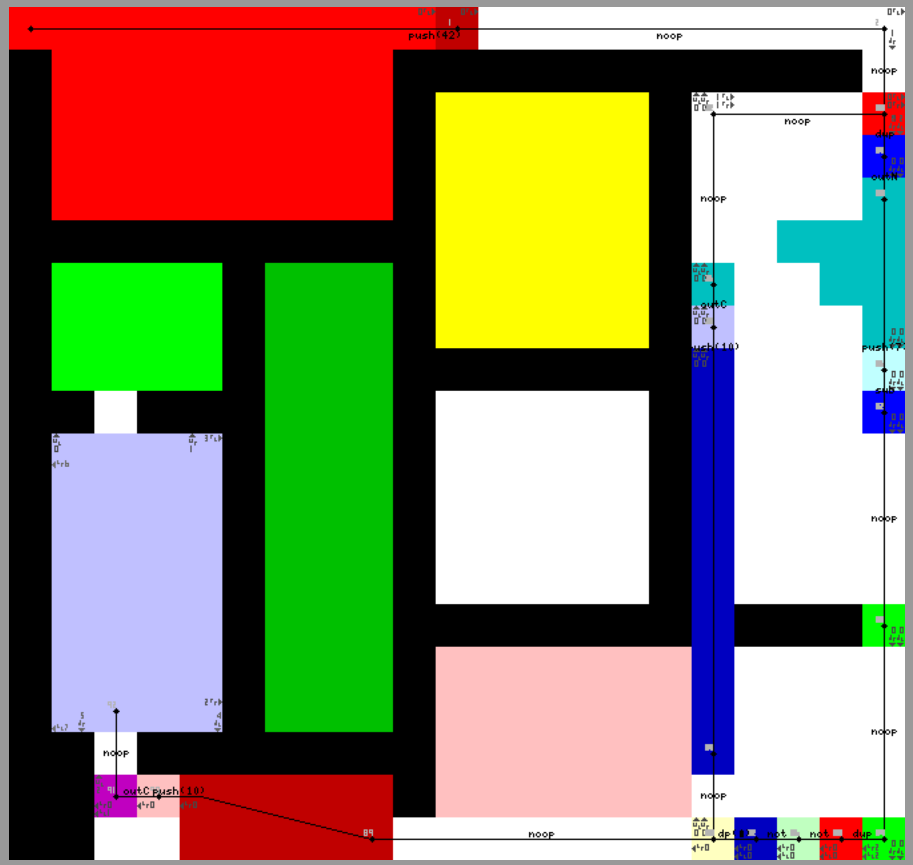


Figure: Trace of a Piet program.

1
14

Stack

command:
pointer()

output:
42
35
28
21

Execution Trace

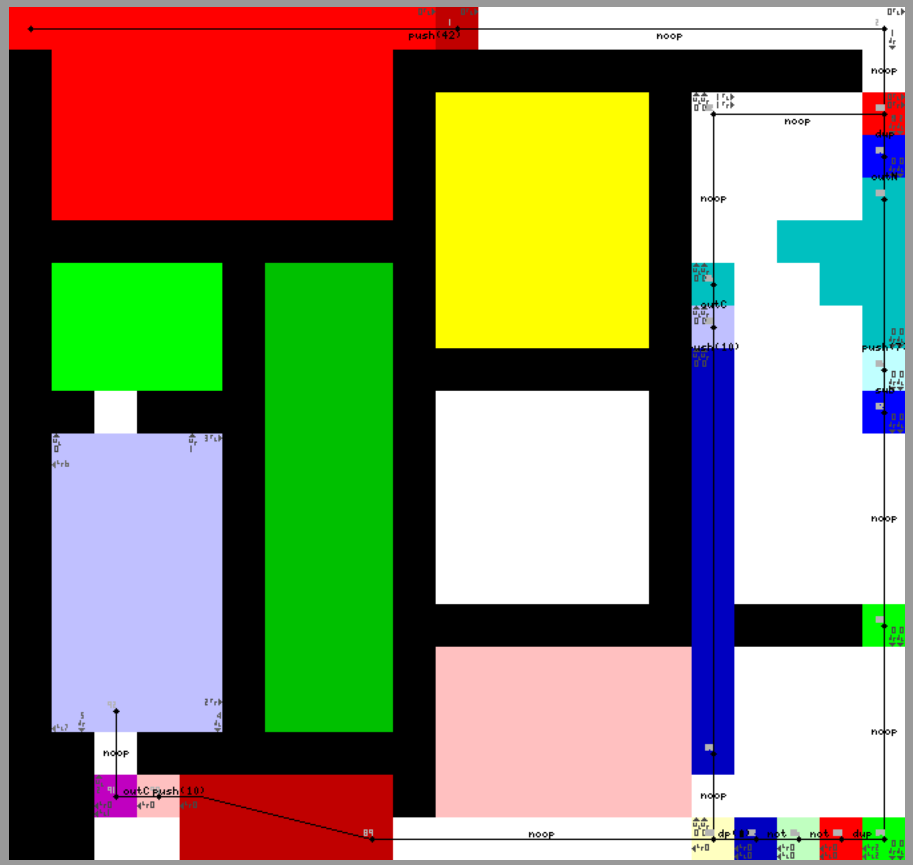
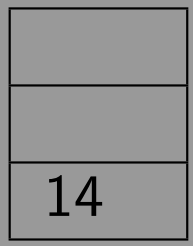


Figure: Trace of a Piet program.



Stack

command:
push(10)

output:
42
35
28
21

Execution Trace

Figure: Trace of a Piet program.

10
14

Stack

command:
outC()

output:
42
35
28
21

Execution Trace

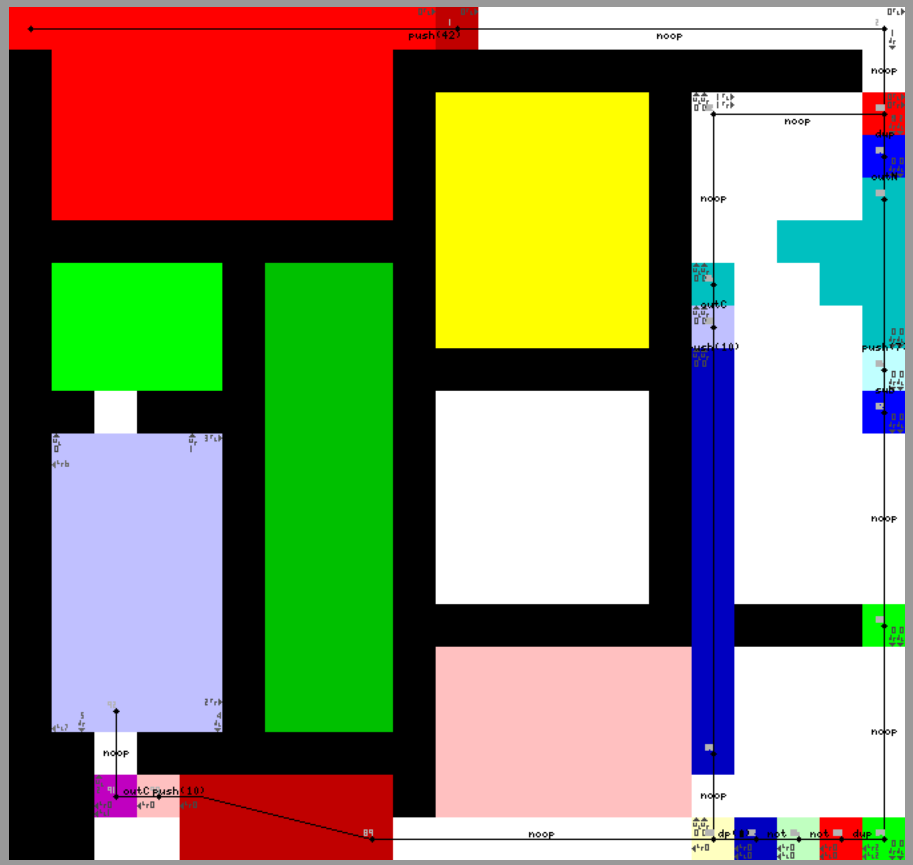
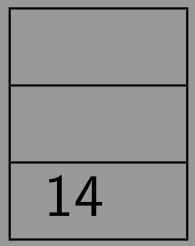


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35
28
21

Execution Trace

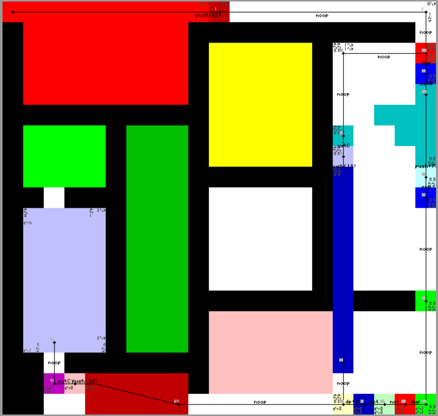


Figure: Trace of a Piet program.

14
14

Stack

command:
outN()

output:
42
35
28
21

Execution Trace

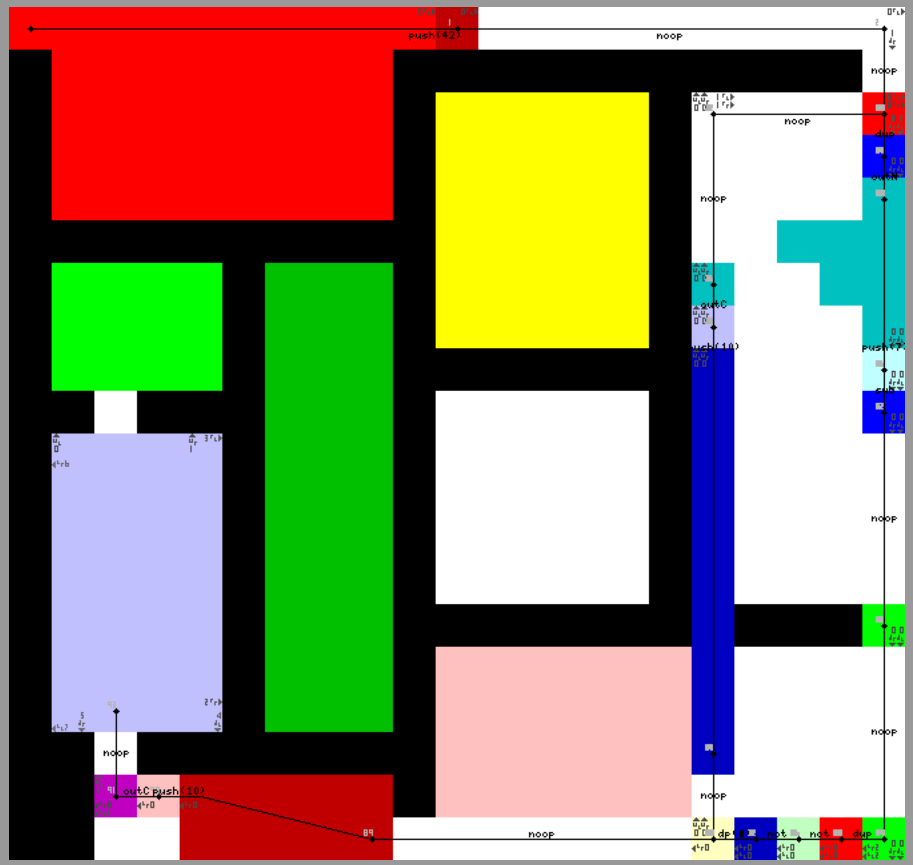
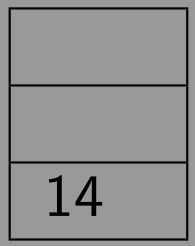


Figure: Trace of a Piet program.



Stack

command:

output:

42
35
28
21
14

Execution Trace

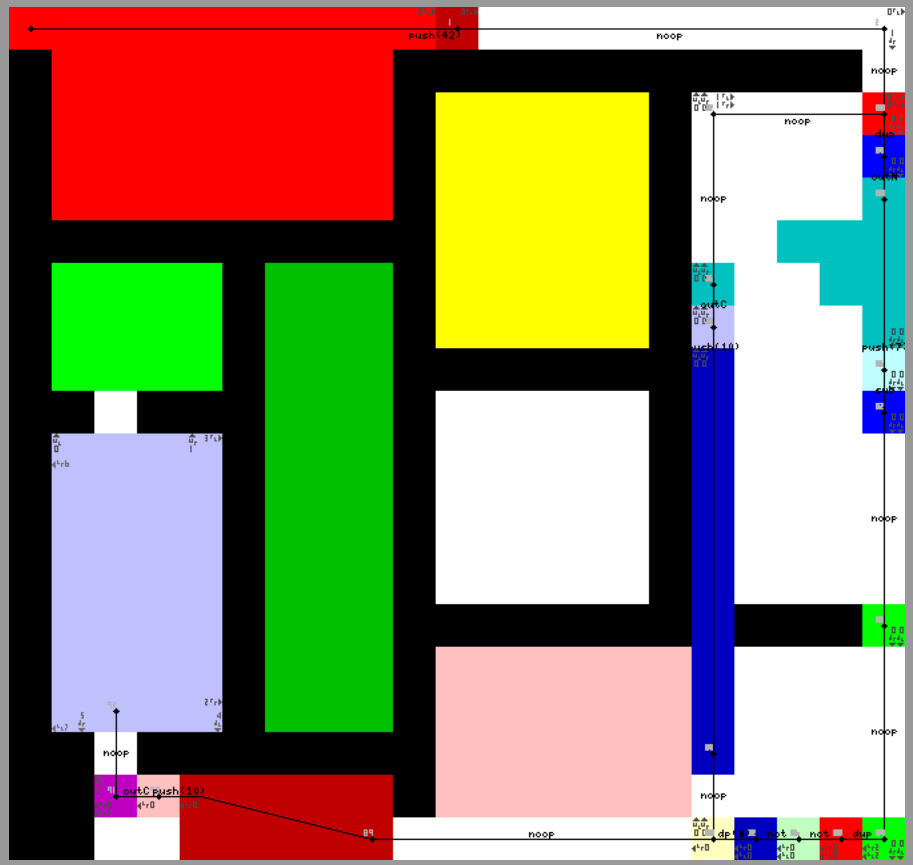
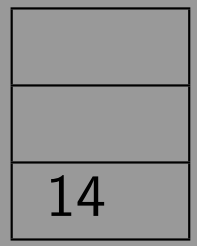


Figure: Trace of a Piet program.



Stack

command:
push(7)

output:
42
35
28
21
14

Execution Trace

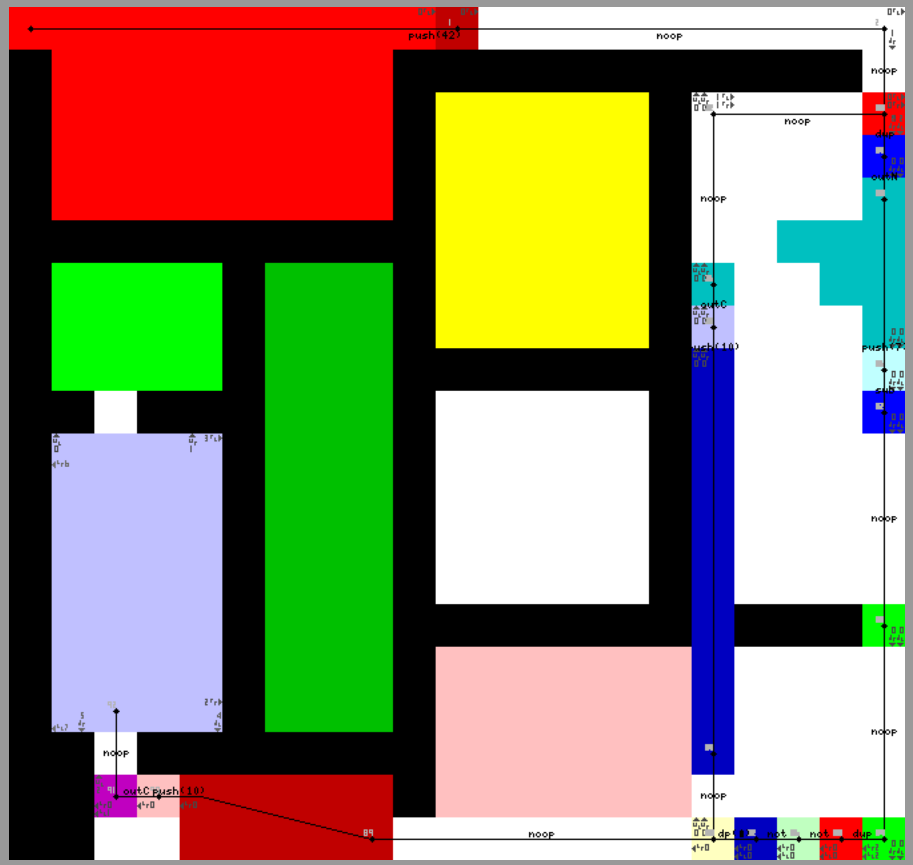


Figure: Trace of a Piet program.

7
14

Stack

command:
sub()

output:
42
35
28
21
14

Execution Trace

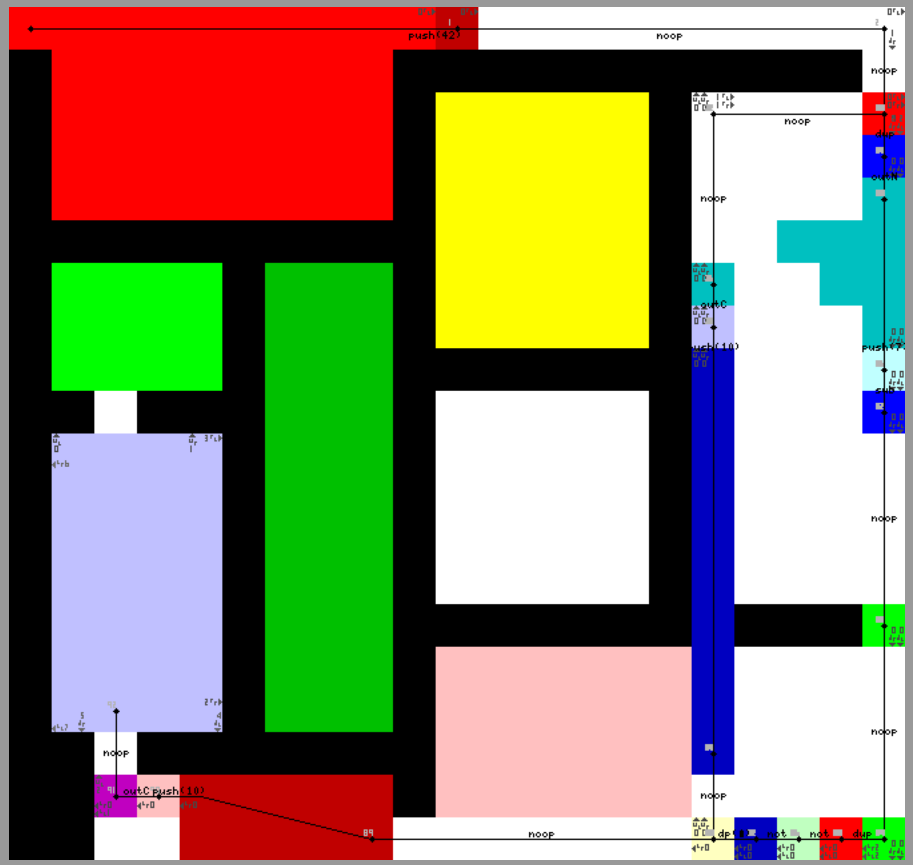
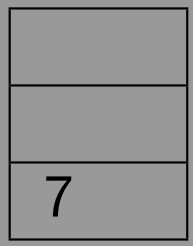


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35
28
21
14

Execution Trace

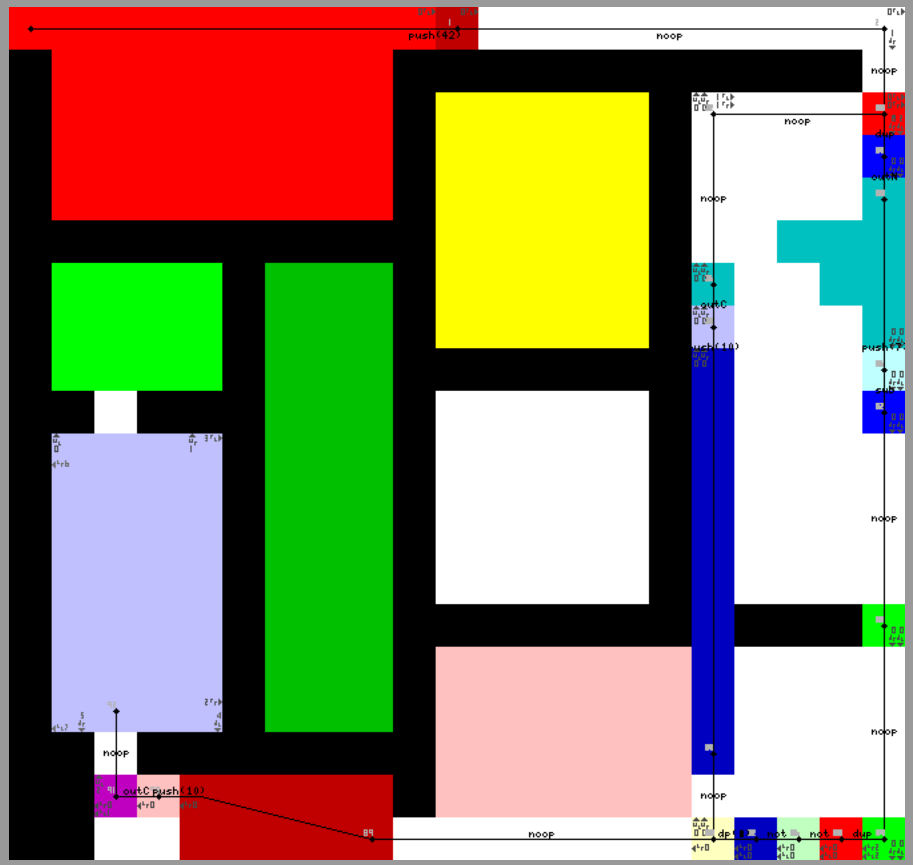


Figure: Trace of a Piet program.

7
7

Stack

command:
not()

output:
42
35
28
21
14

Execution Trace

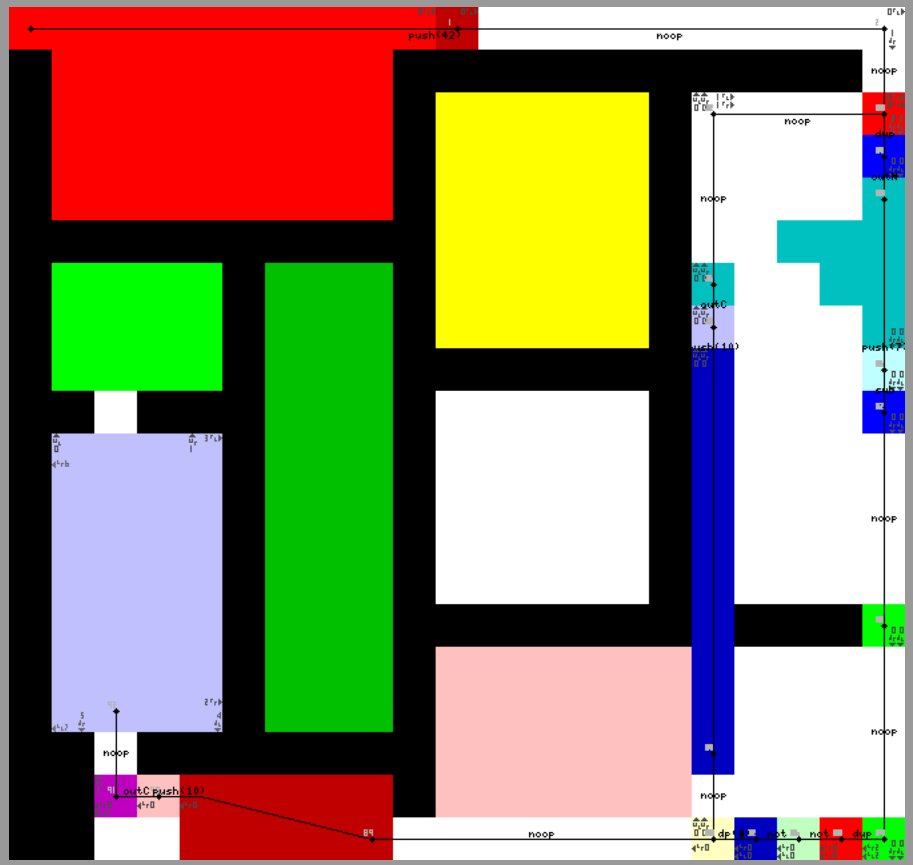


Figure: Trace of a Piet program.

0
7

Stack

command:
not()

output:
42
35
28
21
14

Execution Trace

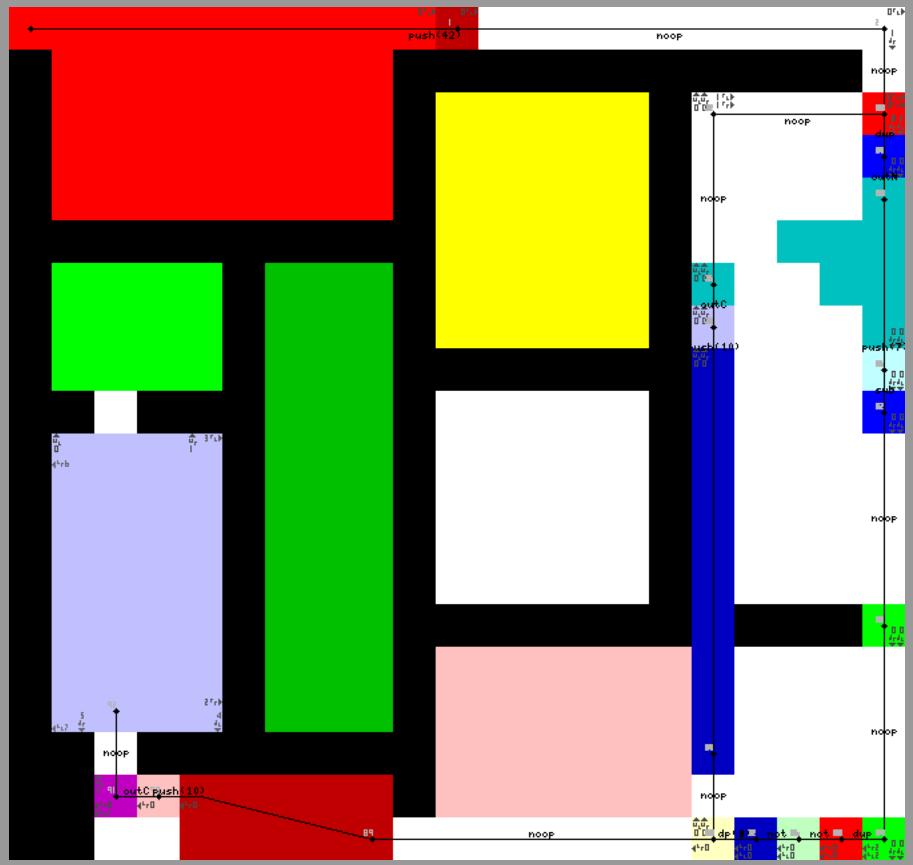


Figure: Trace of a Piet program.

1
7

Stack

command:
pointer()

output:
42
35
28
21
14

Execution Trace

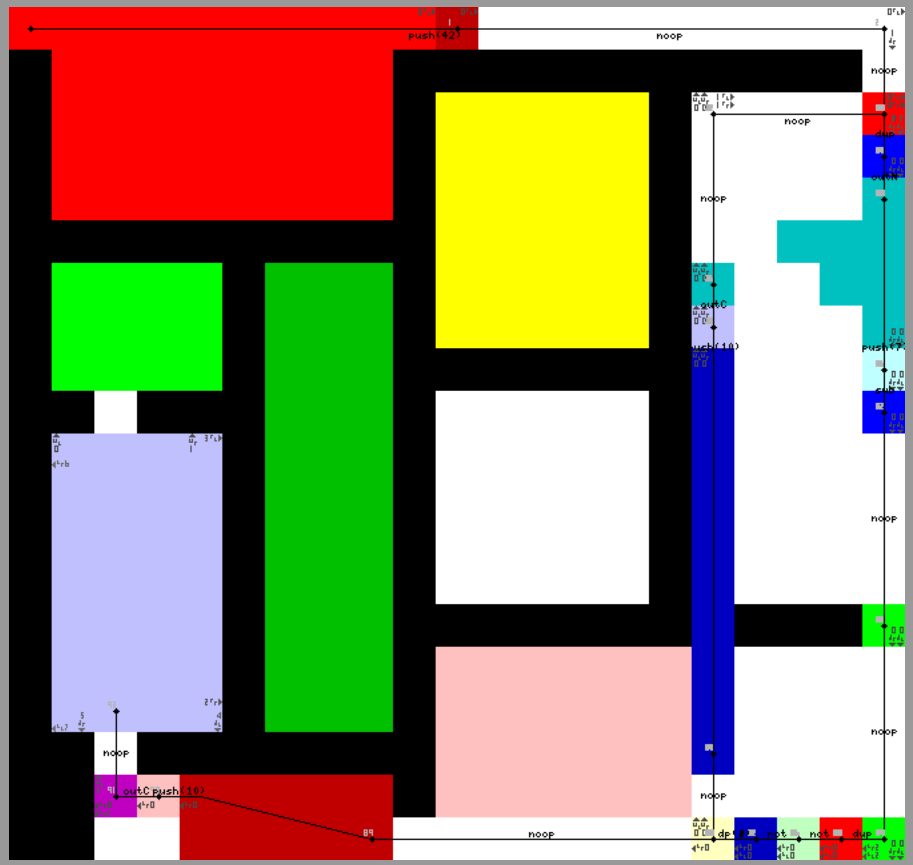
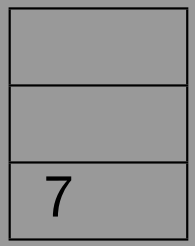


Figure: Trace of a Piet program.



Stack

command:
push(10)

output:

42
35
28
21
14

Execution Trace

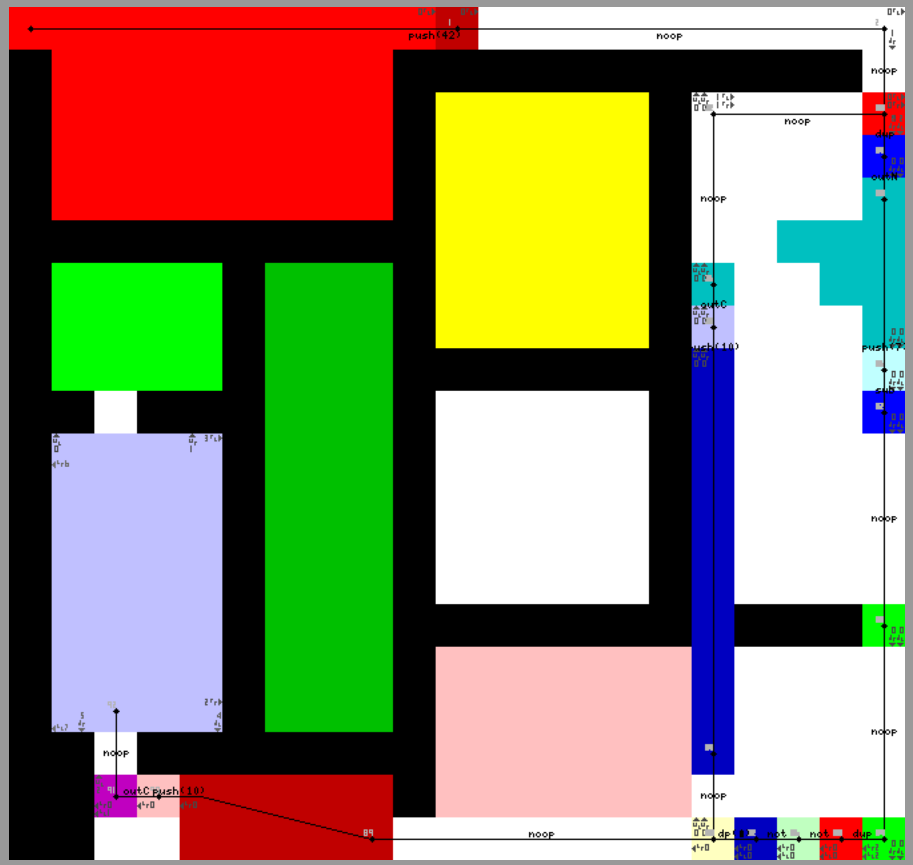


Figure: Trace of a Piet program.

10
7

Stack

command:
outC()

output:
42
35
28
21
14

Execution Trace

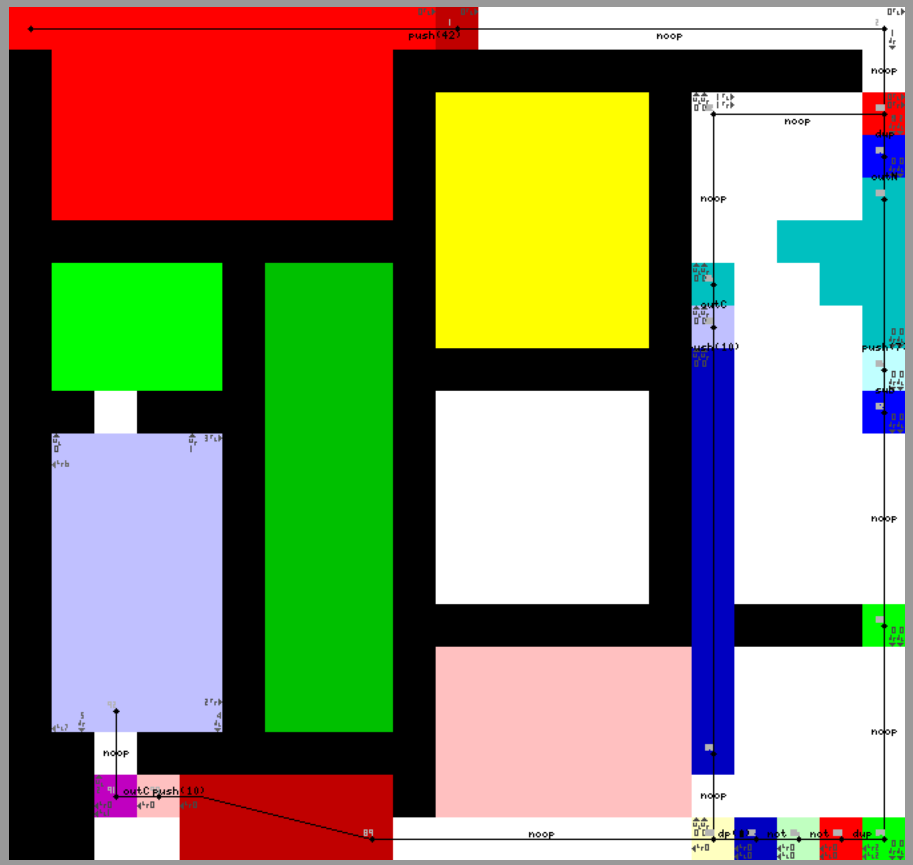
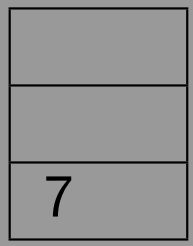


Figure: Trace of a Piet program.



Stack

command:
dup()

output:
42
35
28
21
14

Execution Trace

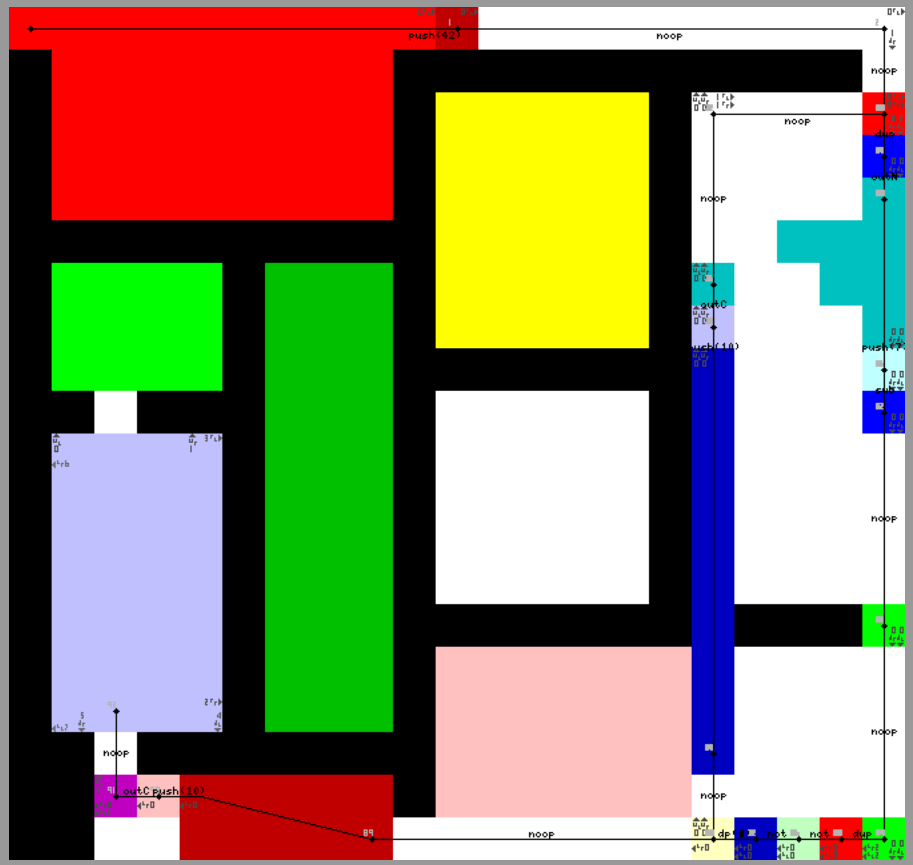


Figure: Trace of a Piet program.

7
7

Stack

command:
outN()

output:
42
35
28
21
14

Execution Trace

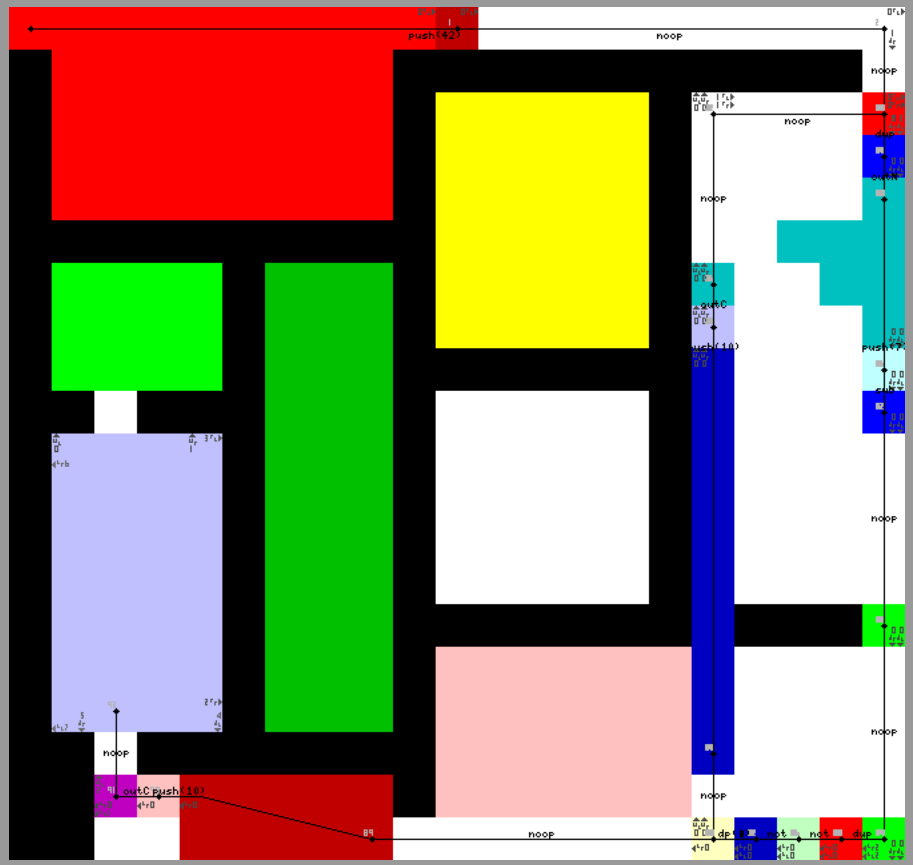
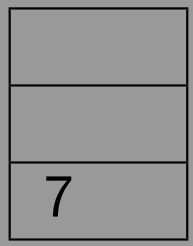


Figure: Trace of a Piet program.



Stack

command:

output:

42
35
28
21
14
7

Execution Trace

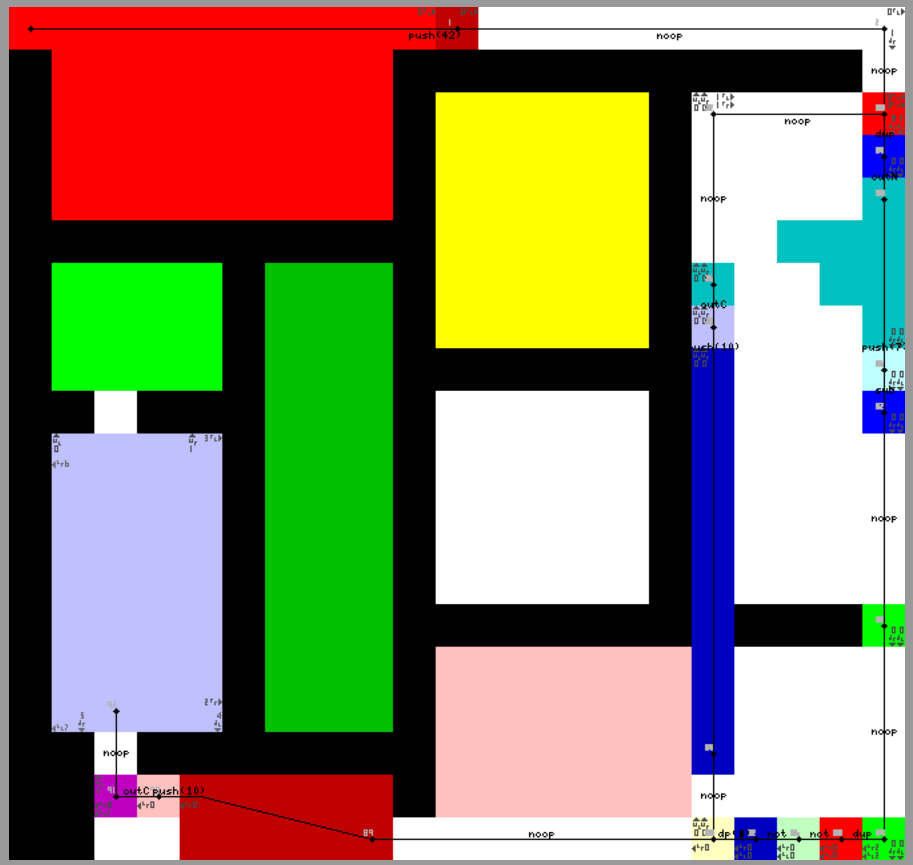
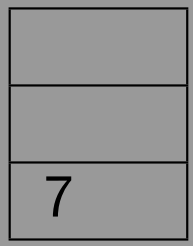


Figure: Trace of a Piet program.



Stack

command:
push(7)

output:

42
35
28
21
14
7

Execution Trace

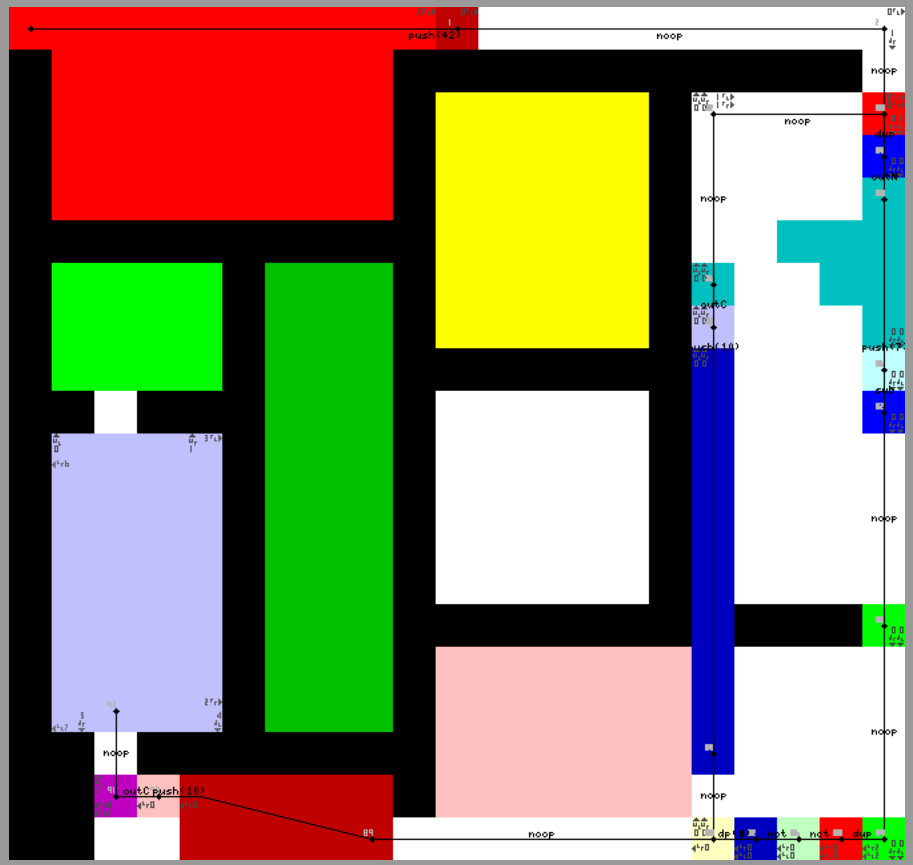


Figure: Trace of a Piet program.

7
7

Stack

command:
sub()

output:
42
35
28
21
14
7

Execution Trace

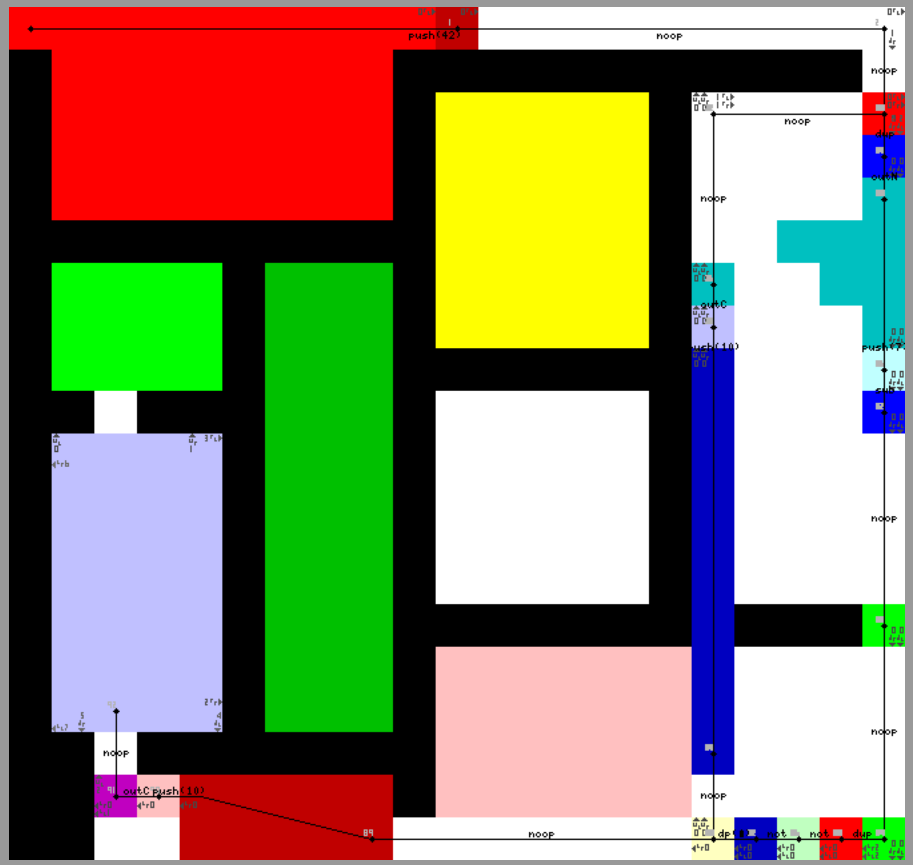
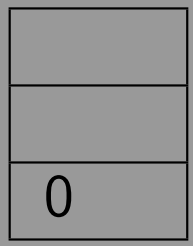


Figure: Trace of a Piet program.



Stack

command:
dup()

output:

42
35
28
21
14
7

Execution Trace

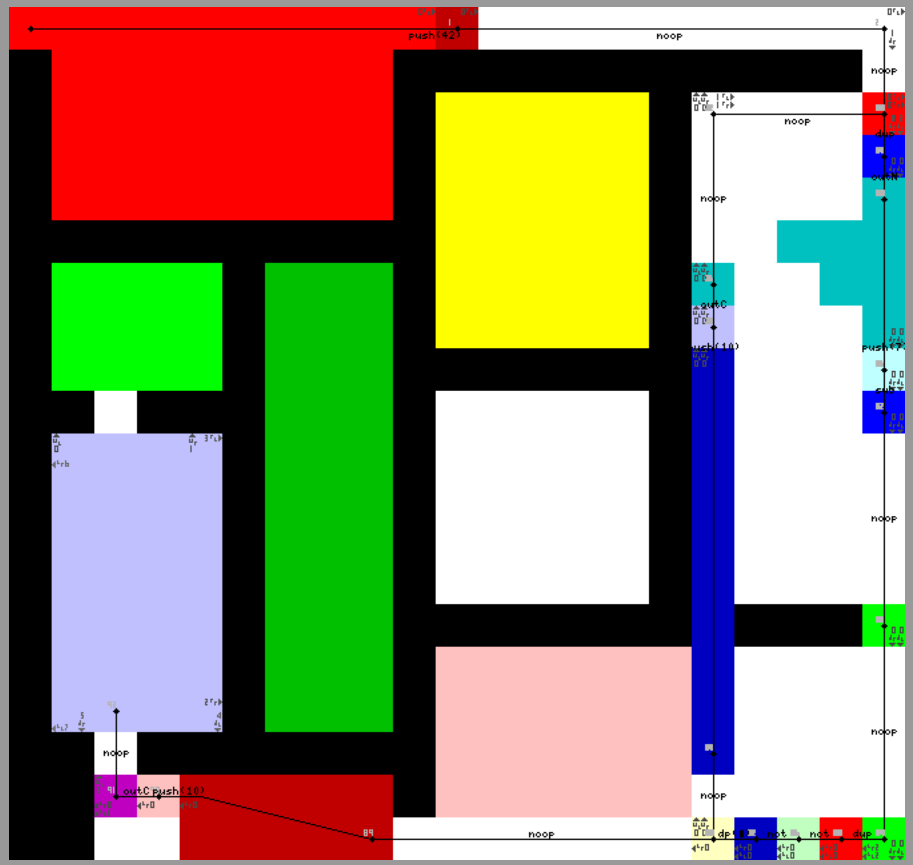


Figure: Trace of a Piet program.

0
0

Stack

command:
not()

output:
42
35
28
21
14
7

Execution Trace

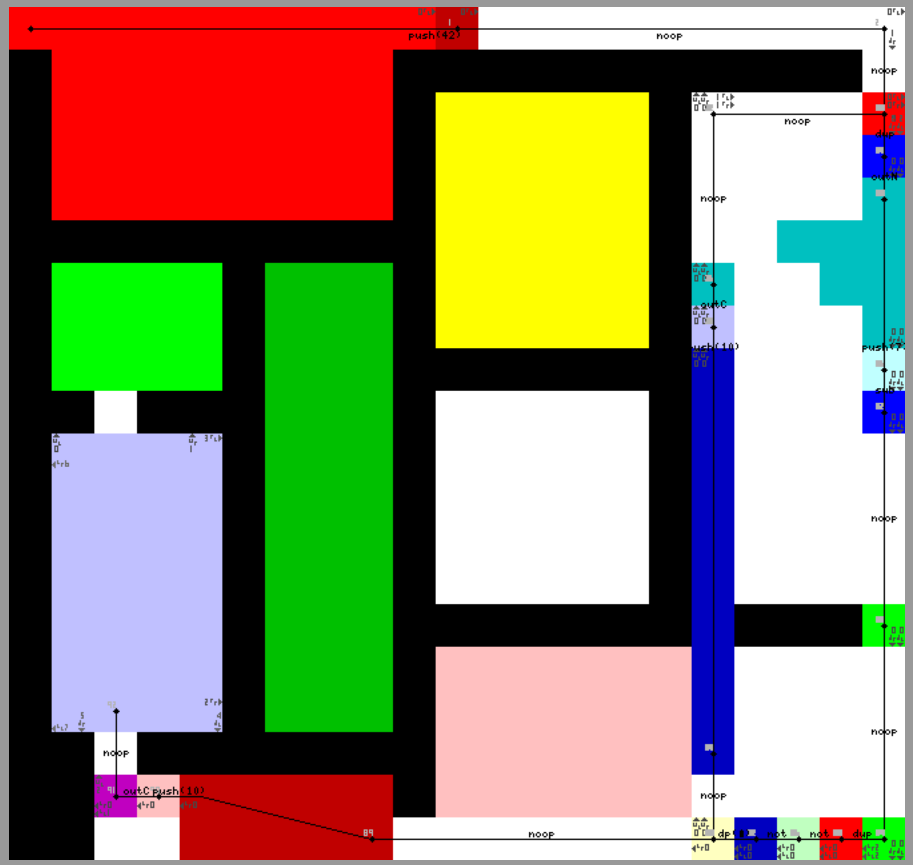


Figure: Trace of a Piet program.

1
0

Stack

command:
not()

output:
42
35
28
21
14
7

Execution Trace

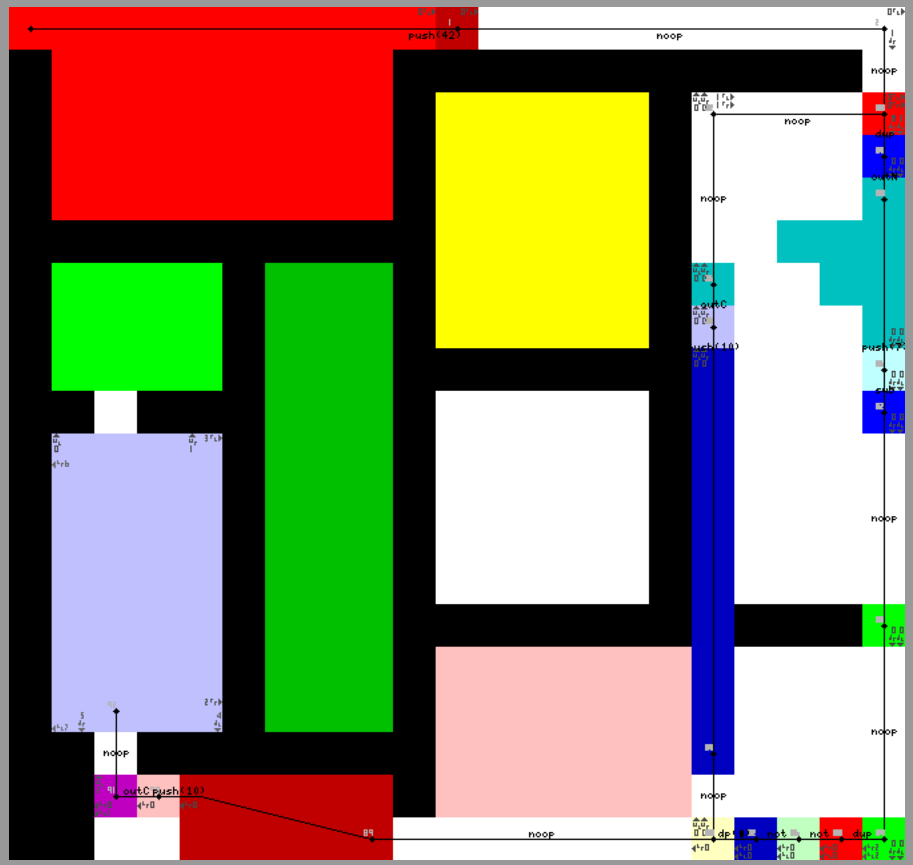


Figure: Trace of a Piet program.

0
0

Stack

command:
pointer()

output:
42
35
28
21
14
7

Execution Trace

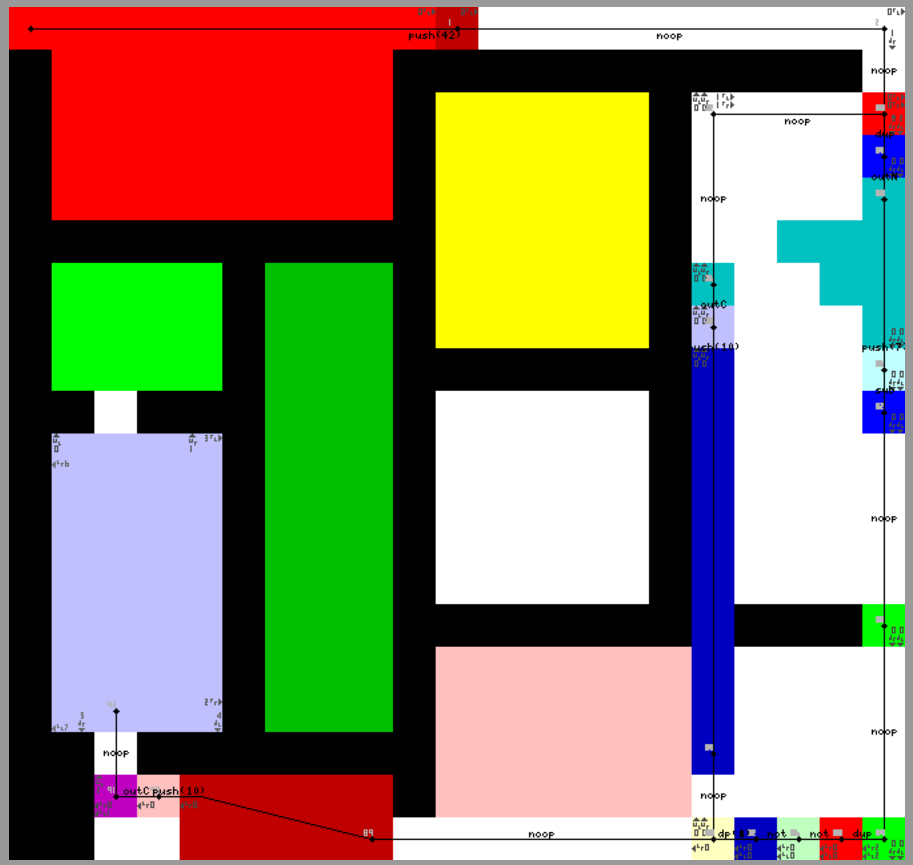
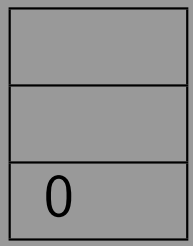


Figure: Trace of a Piet program.



Stack

command:
push(10)

output:

42
35
28
21
14
7

Execution Trace

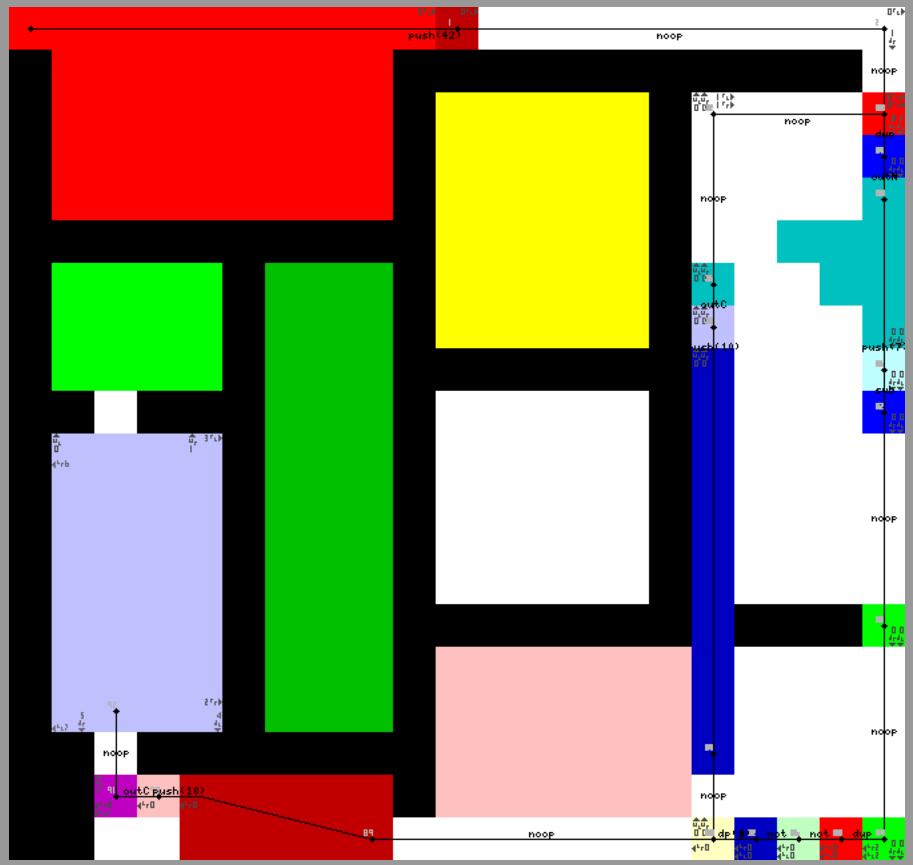


Figure: Trace of a Piet program.

10
0

Stack

command:
outC()

output:

42
35
28
21
14
7

Execution Trace

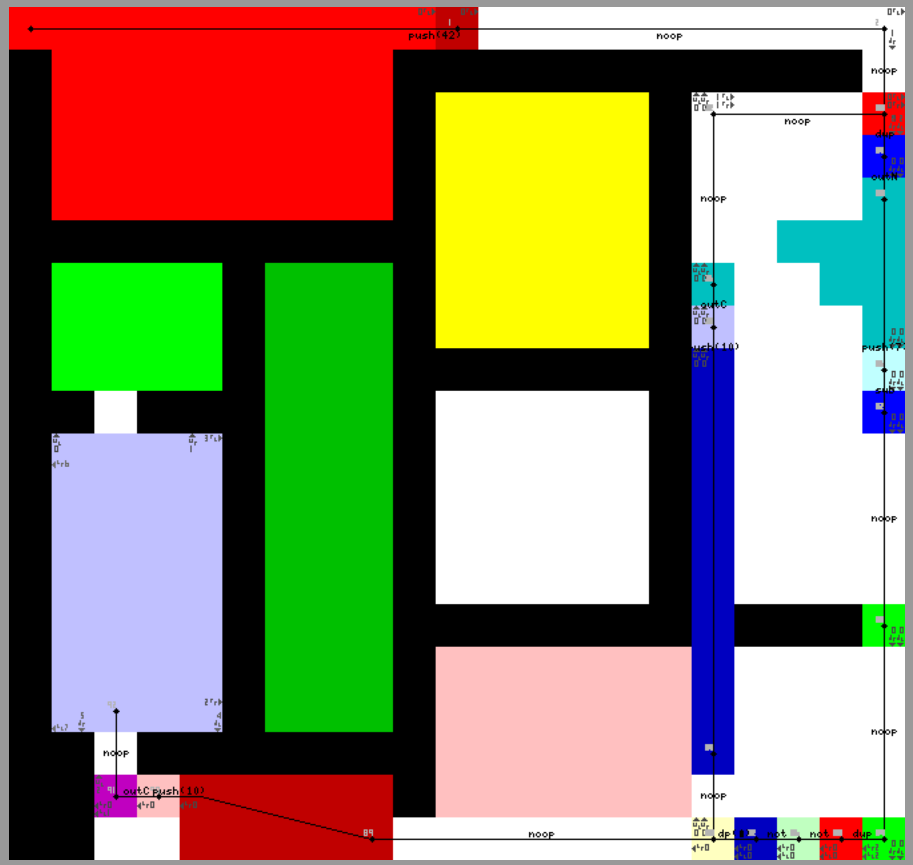
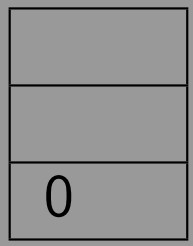


Figure: Trace of a Piet program.



Stack

command:
terminate

output:

42
35
28
21
14
7

Language Specification I [3]

Building Blocks

- **Codel:** single pixel or analogous
- **Color Block:** continuous block of codels of same color, number of codels in one color block encode integer-values
- **Stack:** structure to save integer values
- **Direction Pointer (DP):** direction of next traversal step, pointing to the right at start (right, left, up, down)
- **Codel Chooser (CC):** looking-direction of next step, looking to left at start (left, right)

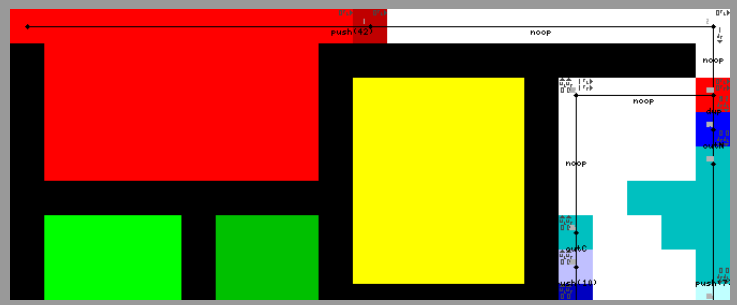


Figure: Partial trace of a Piet program, starts in top left codel.

Language Specification II [3]

Used Colors

light	red	yellow	green	cyan	blue	magenta
normal	red	yellow	green	cyan	blue	magenta
dark	red	yellow	green	cyan	blue	magenta
	white			black		

Used Cycles

- Hue Cycle:
  red $\rightarrow$ yellow $\rightarrow$ green $\rightarrow$ cyan $\rightarrow$ blue $\rightarrow$ magenta $\rightarrow$ red
- Lightness Cycle:
  light $\rightarrow$ normal $\rightarrow$ dark $\rightarrow$ light

Language Specification III [3]

Commands

- Execution when moving from one color block to another
- Command look-up with amount of steps in color- and lightness-cycle

	Change in Lightness		
Change in Hue	None	1 darker	2 darker
none		push	pop
1 step	add	subtract	multiply
2 steps	divide	mod	not
3 steps	greater	pointer	switch
4 steps	duplicate	roll	in(number)
5 steps	in(char)	out(number)	out(char)

Figure: Available commands in Piet.

Npiet

npiet [2]

- Developed by Erik Schoenfelder
- Command-line interpreter (ppm, png and gif)
- Graphical program editor

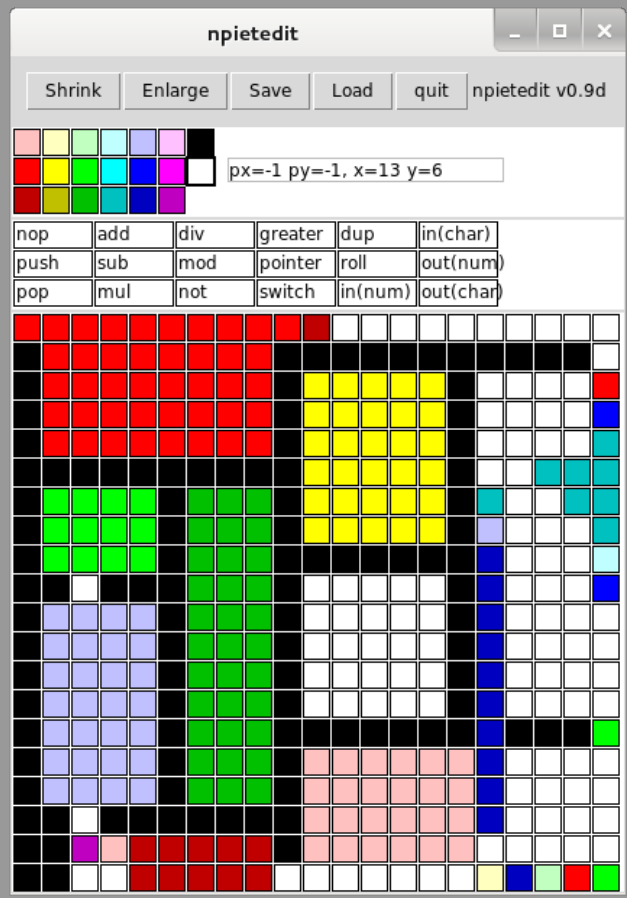


Figure: Graphical Piet editor *npietedit*.

Demo

Npiet Live Demo!

Examples

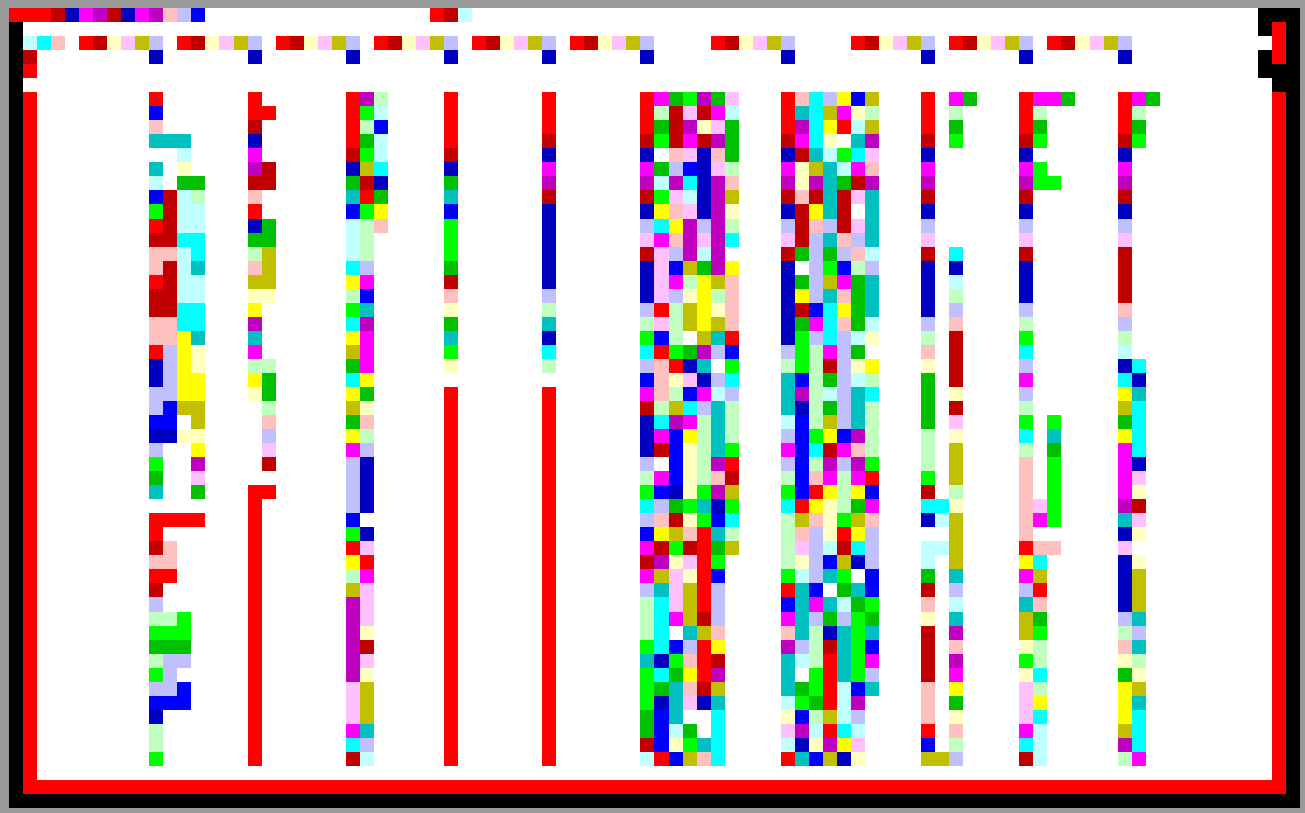


Figure: Prints song lyrics of “99 bottles of beer”. [3]

Examples

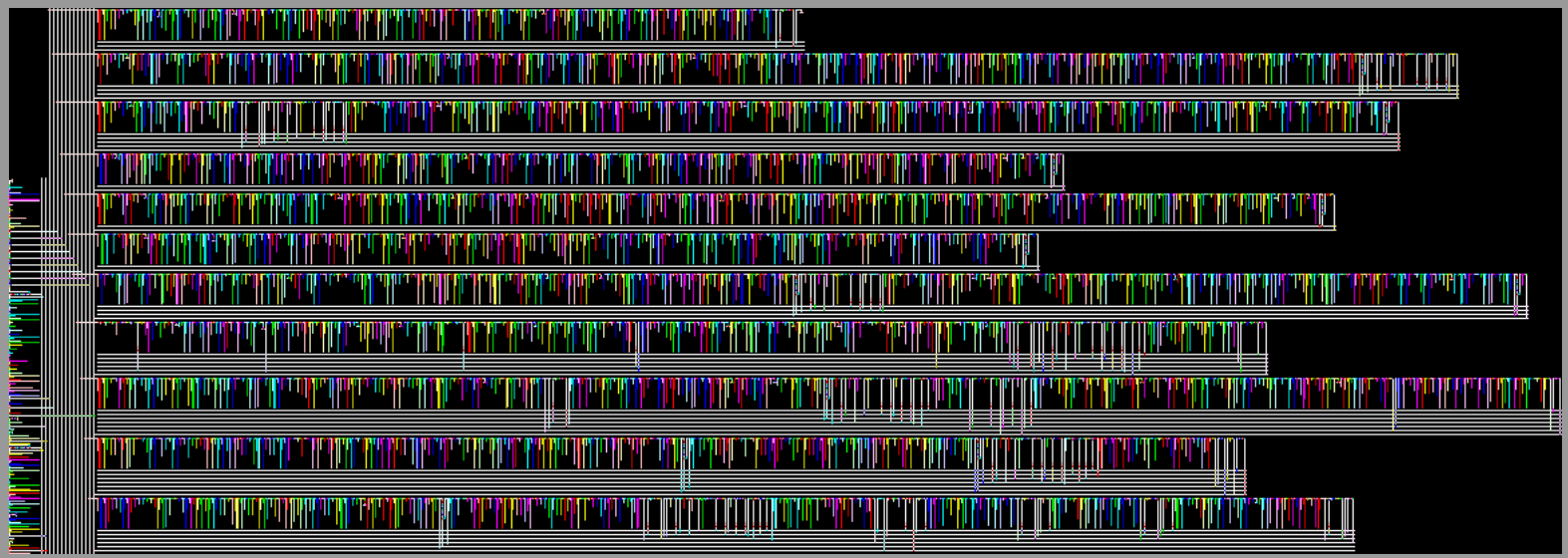


Figure: Text-adventure in Piet (rotated 90° counter-clock-wise). [3]

Examples

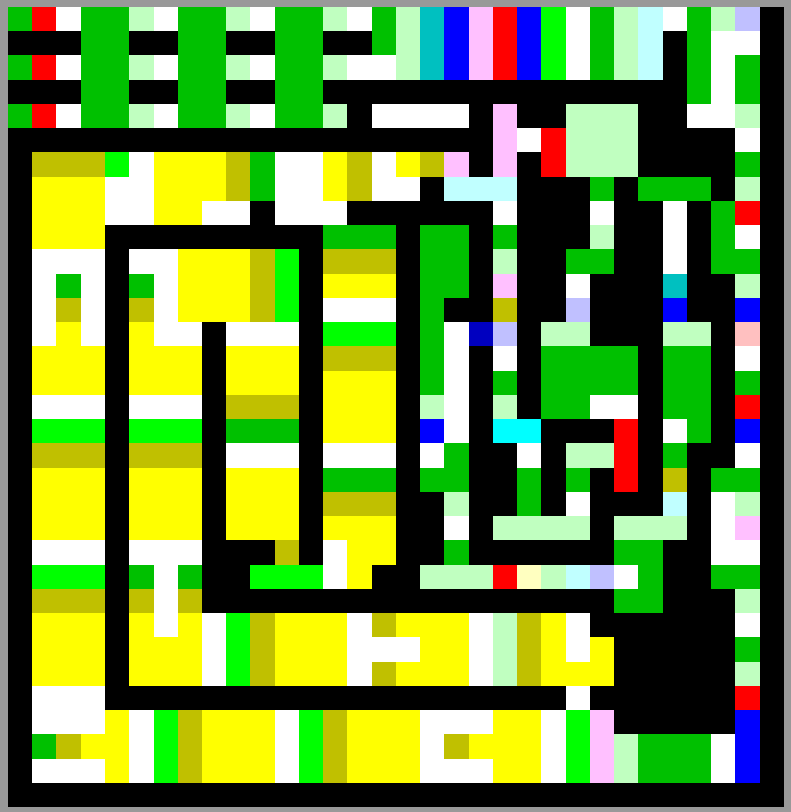


Figure: Calculates the day of the week, given year, month and day as input. [3]

Examples

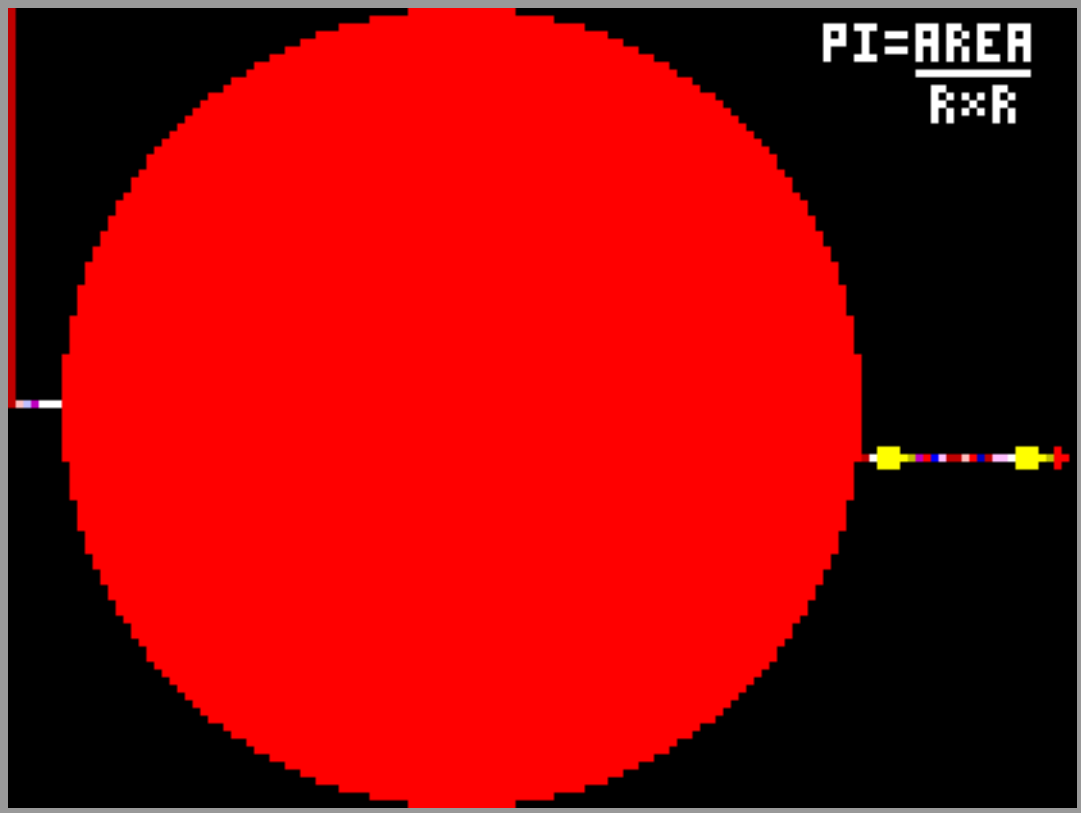


Figure: Approximates π as integer value (Output: 31405). [3]

Examples

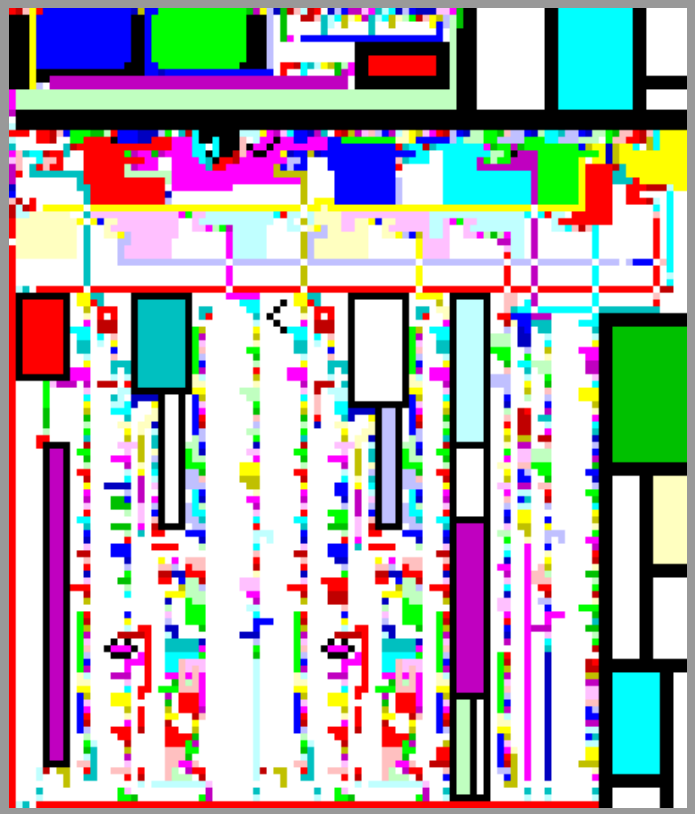


Figure: Interpreter for Brainfuck. [3]

Characteristics

- Esoteric language
- Interpreted
- Stack oriented
- "... very likely Turing complete" [4] [1]

References and Usage [3]

Known uses and occurrence of Piet

- Online articles
- MIT Mystery Hunt 2002
- “50 in 50” talk about history of computer languages from Guy L. Steele (Lisp, Scheme, Emacs,...) and Richard P. Gabriel (Lisp, Lisp Benchmarks, ...)
- Small community around the world
- Subject in academic education as example for esoteric languages (e.g. in specialization seminars)

Links and Bibliography I


mamememo - blog of Yusuke Endoh.
`http://mamememo.blogspot.co.at/2009/10/piet-is-turing-complete.html`.
Accessed 2015-05-31.


npiet - an interpreter and editor for the piet programming language.
`http://www.bertnase.de/npiet/`.
Accessed: 2015-05-29.


Piet.
`http://www.dangermouse.net/esoteric/piet/`.
Accessed: 2015-05-29.

Links and Bibliography II

Piet - Esolang.
`https://esolangs.org/wiki/Piet`.
Accessed 2015-05-31.

Susie Hodge.
Neoplastizismus.
In *50 Schlüsselideen Kunst*, pages 128–131. Springer Berlin Heidelberg, 2014.