Self and OpenSmalltalk VMs

Jecel Mattos de Assumpção Jr
jecel@merlintec.com

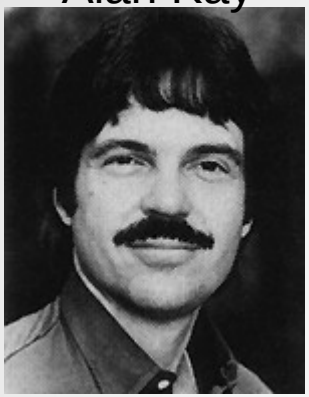
RISC-V J Extension Meeting
September 20, 2018

outline

- Smalltalk history and hardware
- Self virtual machine
- OpenSmalltalk virtual machine

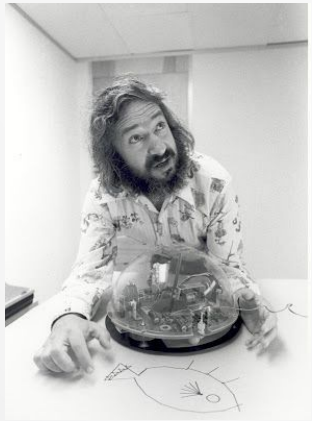
Smalltalk history

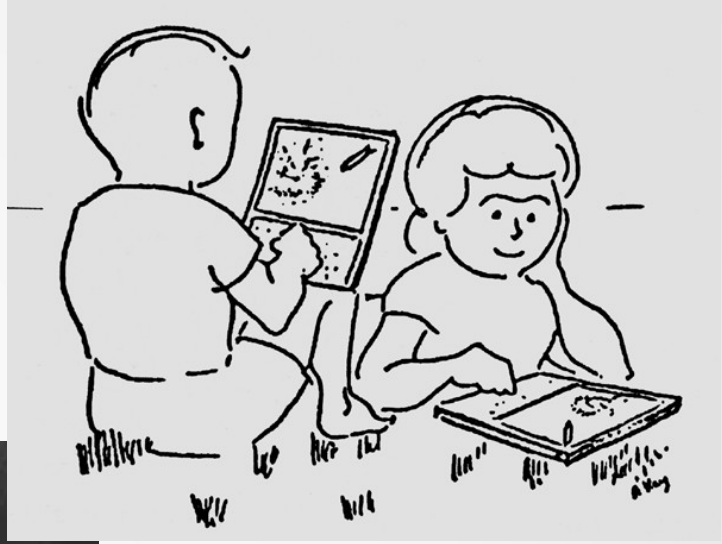
Alan Kay



Flex 1968/1969

Seymour Papert





Dynabook

B220 tape format

Simula I

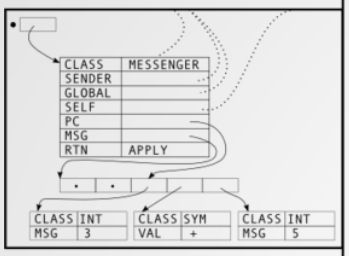
LISP 1.5

Ivan Sutherland's Sketchpad

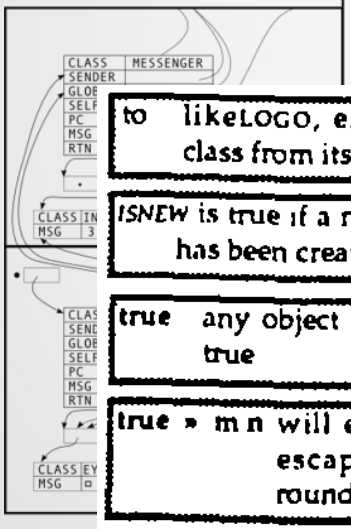
Smalltalk-72: one page design

The "One Pager"

"Before"



"During"




```
e       (the environment) will be bound to the current Messenger object
result  holds the result of a send, usually to be *applied* to next part of message

eval:   if null(e.MSG)       then 'result <- nil; goto apply;
        if escape(e.MSG)     then goto escapes;
        if atom(e.MSG)       then 'result <- lookupvalue(e, e.MSG); goto apply;
        if notlist(e.MSG)    then 'result <- e.MSG; goto apply;

evlist: 'e <- Table(CLASS,  MESSENGER,
                    SENDER, e,
                    GLOBAL, e.GLOBAL,
                    zSELF,  e.SELF,
                    PC,     1,
                    MSG,    e.MSG.PC
                    RTN,    APPLY);
            goto eval;

apply: 'e <- e.SENDER;
           e.PC <- e.PC +1;
           if e.PC > length(MSG) then goto dispatchrtn;
           if e.MSG.PC = ',  then e.PC <- e.PC + 1; goto evlist;
           if e.MSG.PC = '.  then if result = 'false
                                  then e.PC <- e.PC + 2; goto evlist;
                                  else e.PC <- e.PC + 1;
                                       'e <- Table(CLASS,  MESSENGER,
                                                   SENDER, e,
                                                   GLOBAL, GLOBAL,
                                                   SELF,   result,
                                                   PC,     1,
```

```
to ^ b (:b{?} metacodefor('return <- e.b; goto apply))
```

Dan Ingalls

DG Nova Basic => Nova assembly

to likeLOGO, except makes a class from its message

ISNEW is true if a new instance has been created

true any object not false acts as true

true » m n will evaluate m and escape from surrounding ( )

false » m n will evaluate n

: evals the next part of message and binds result to the variable in its message

temporary variable

Instance variables

```
to Pair b | h t                          "b is temp. h, t are internal instance vars"
    (ISNEW   » (:h. :t)                  "cons—if no explicit return is given, SELF is returned
    □hd      » (□<- » (^:h)^h)           "replaca and car"
    □tl      » (□<- » (:t)^t)            "replacd and cdr"
    □isPair  » (^true)
    □print   » ('(print. SELF mprint)
    □mprint  » (h print. t isNil » (') print) t isPair » (t mprint) » '• print. t print. ') print)
    □length  » (t isPair » (^1+t length) 1))
```

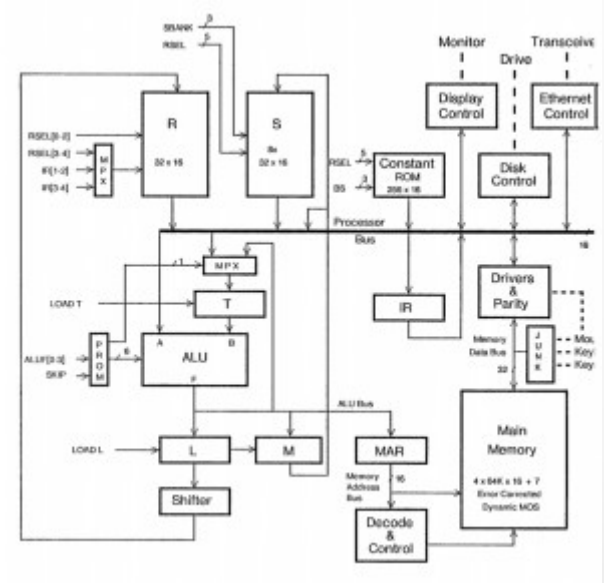
□ eyeball looks to see if its message is a literal token in the message stream

^ send-back returns its value to sender

. "statement separator" value is following message

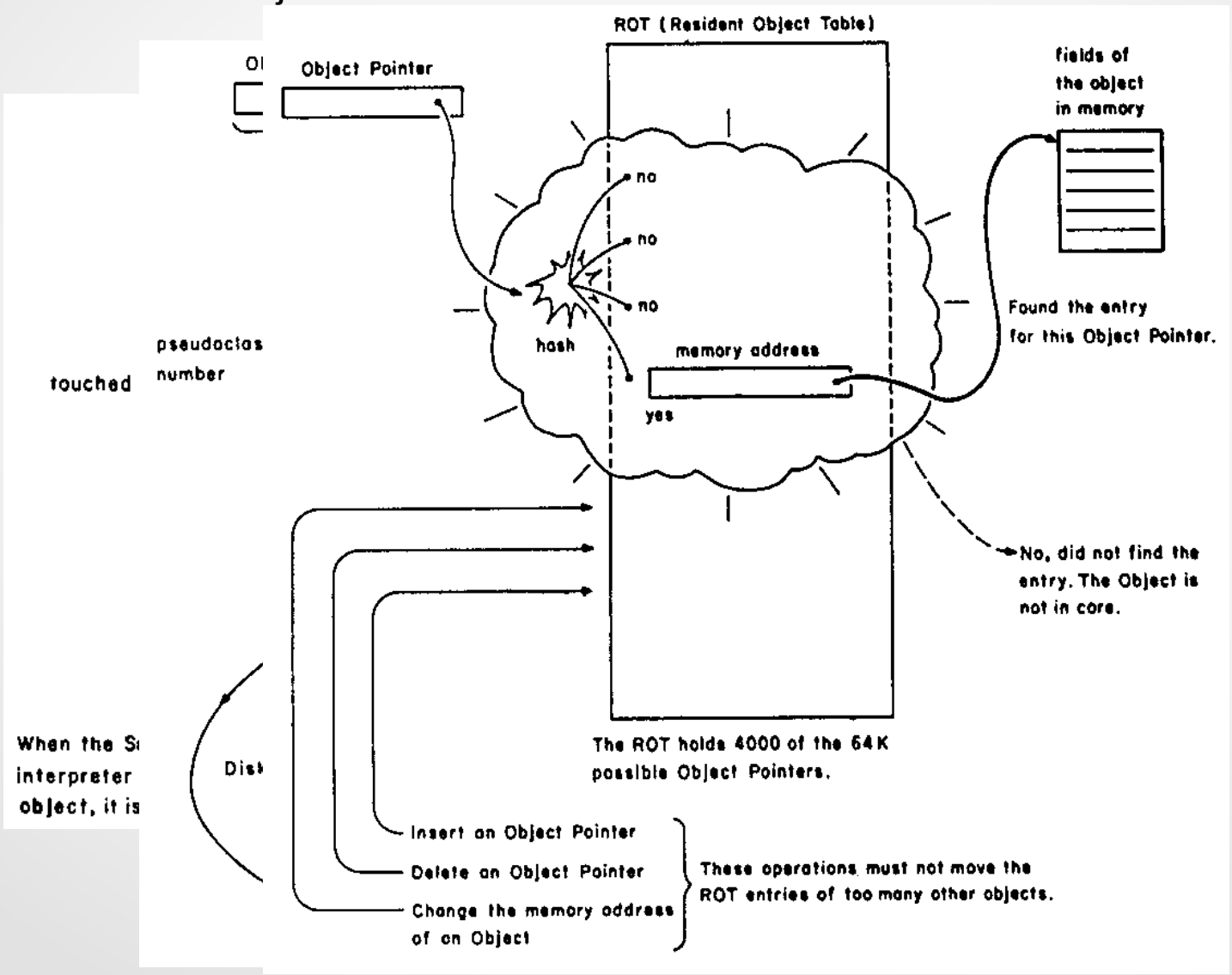
Xerox Alto

Butler Lampson, Chuck Thacker



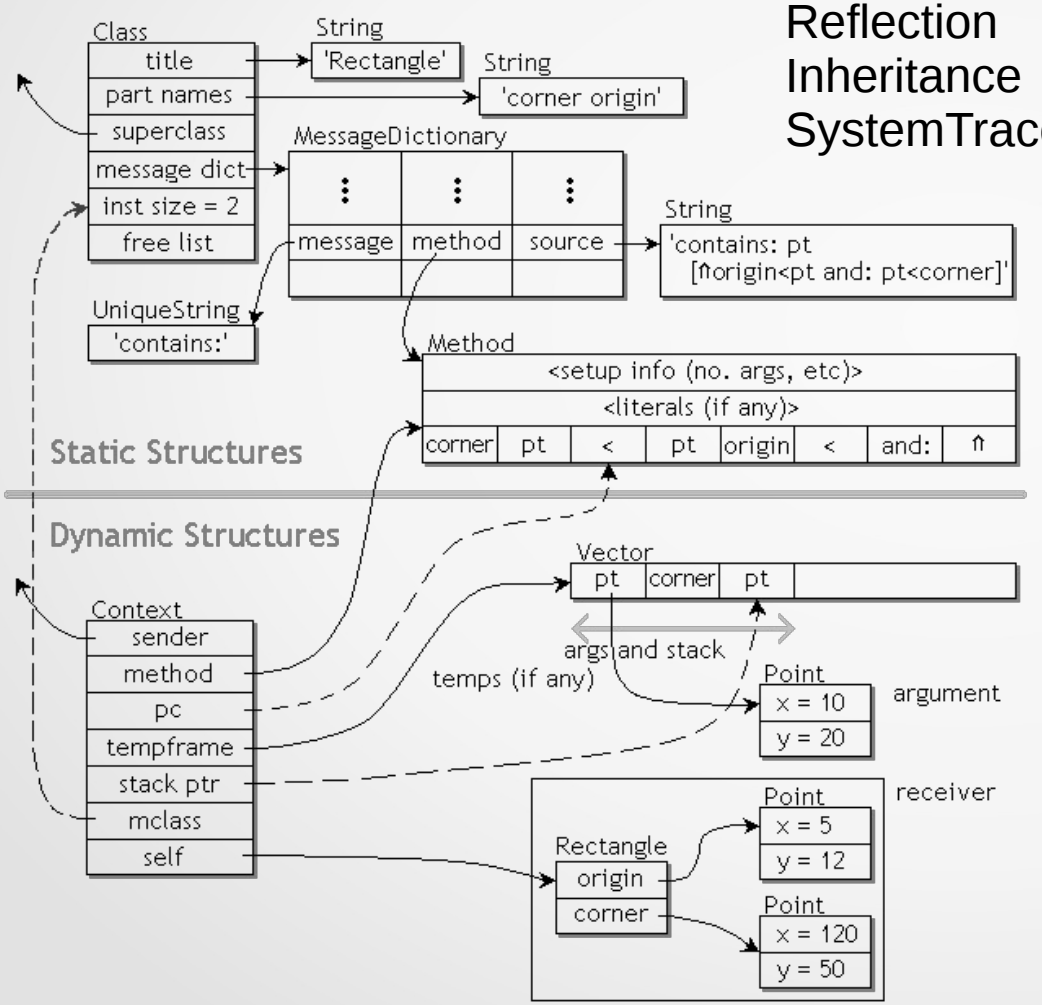
Smalltalk-74 and OOZE

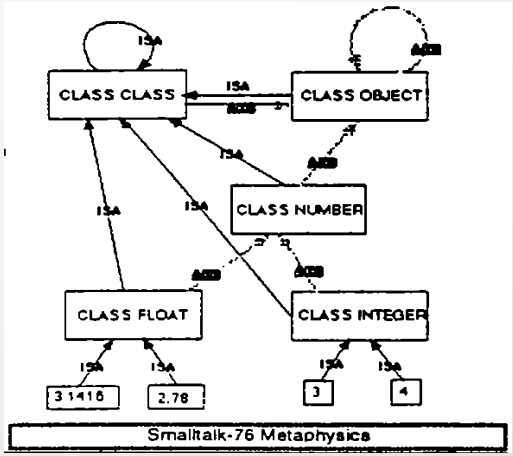
Object-Oriented Zoned Environment



Smalltalk-76

Fixed syntax
Bytecodes
Reflection
Inheritance
SystemTracer

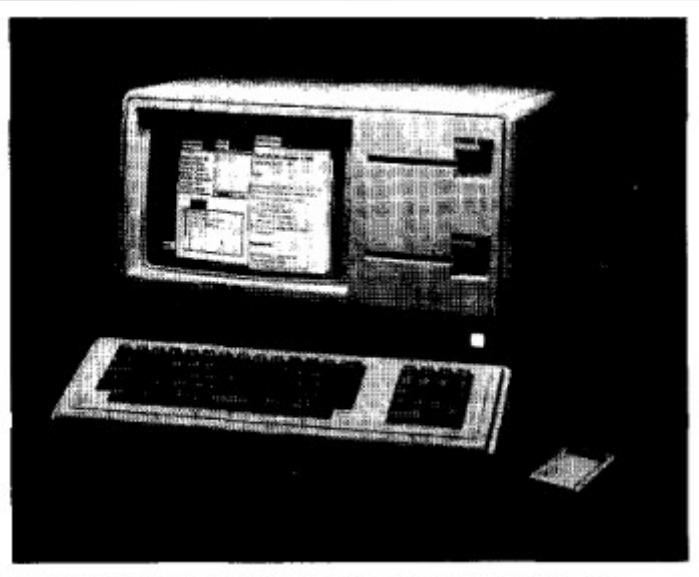




Smalltalk-78 and the Notetaker

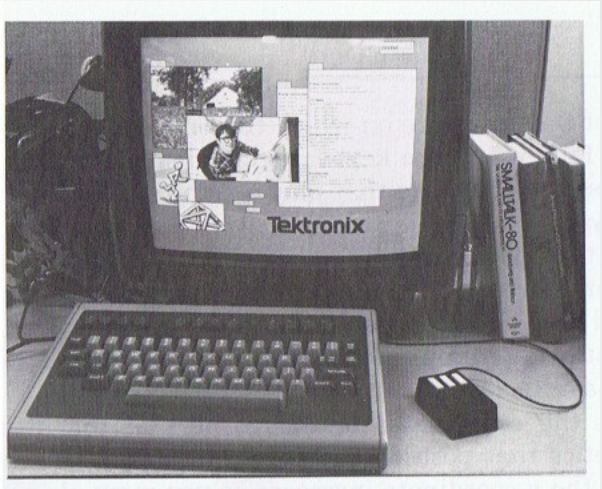
Multiple 8086 processors => kernels
Small integers are unboxed
Object Table

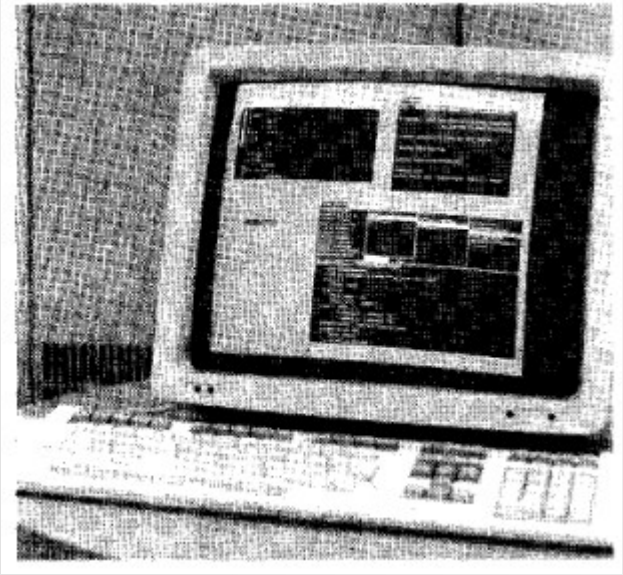
Smalltalk-80



Apple

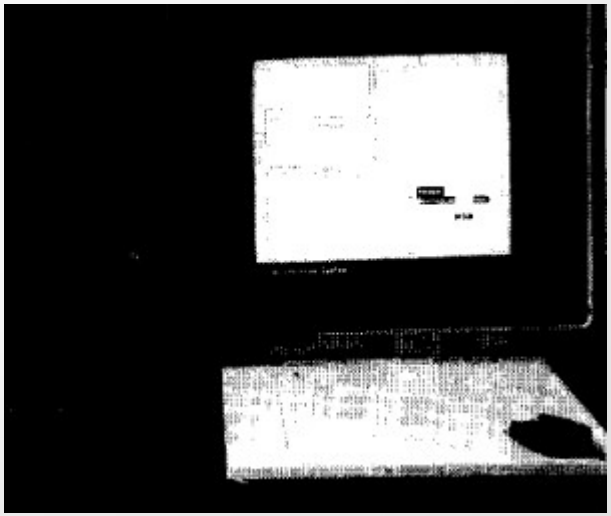
Tektronix



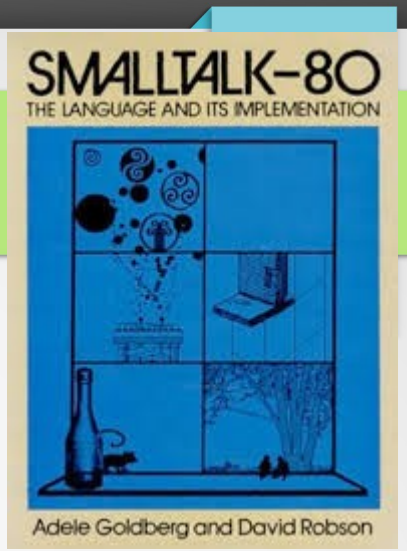


DEC => Berkeley

HP

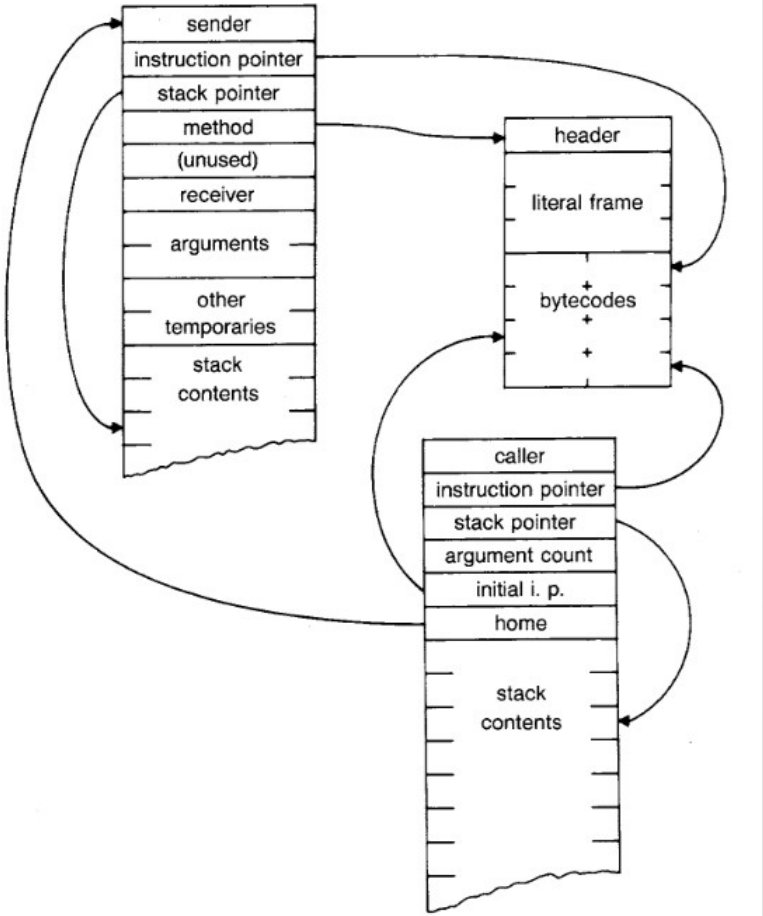


Smalltalk-80



Object Table Index	0
Immediate Signed Integer	1

```
lookupMethodInDictionary: dictionary
  | length index mask wrapAround nextSelector methodArray |
  length ← memory fetchWordLengthOf: dictionary.
  mask ← length−SelectorStart−1.
  index ← (mask bitAnd: (self hash: messageSelector)) + SelectorStart.
  wrapAround ← false.
  [true] whileTrue:
    [nextSelector ← memory fetchPointer: index
                          ofObject: dictionary.
     nextSelector=NilPointer ifTrue: [↑false].
     nextSelector=messageSelector
       ifTrue: [methodArray ← memory fetchPointer: MethodArrayIndex
                                  ofObject: dictionary.
                newMethod ← memory fetchPointer: index−SelectorStart
                                  ofObject: methodArray.
                primitiveIndex ← self primitiveIndexOf: newMethod.
                ↑true].
     index ← index + 1.
     index=length
       ifTrue: [wrapAround ifTrue: [↑false].
                wrapAround ← true.
                index ← SelectorStart]]
```



Hardware: Xerox PARC

1973 1974 1975 1976 1977 1978 1979 1980

Alto

TTL

Alto II

TTL

Dolphin

TTL

Dandelion

2900 bitslice

Dorado

ECL

Hardware: bytecodes

Sword32 (Katana32)

1984
ASIC
University of Tokyo

31 Postive 31 bit integer 0

00....

Object pointer

01....

Stack reference

10....

Negative 31 bit integer

11....

AI32

1986
ASIC
Hitachi

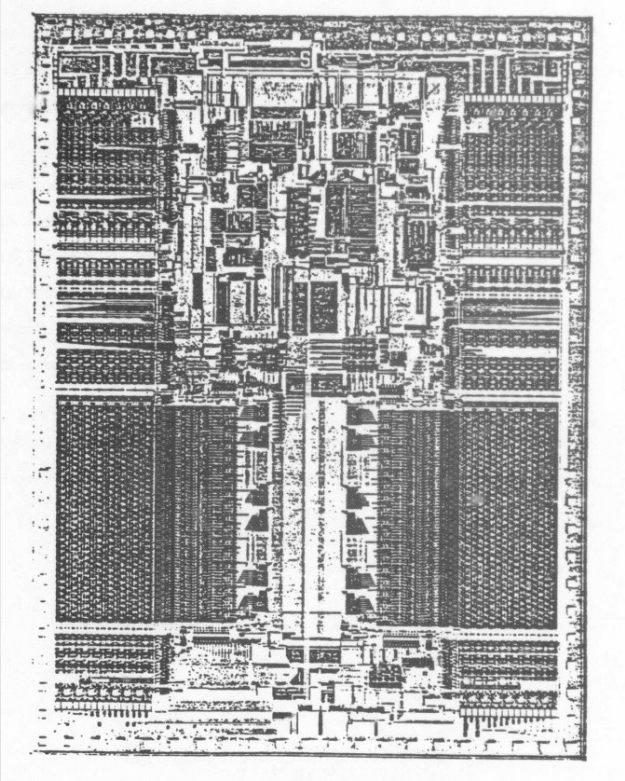
Swamp

1986
Bitslice
University of Toronto

Hardware: RISC III (SOAR)

Smalltalk On A RISC

1983 - 1985



Joan Pendleton, David Ungar, Shing Kong, Will Brown, Frank Dunlap, and Chris Marino

Professors David Hodges and David Patterson

Hardware: RISC III (SOAR)

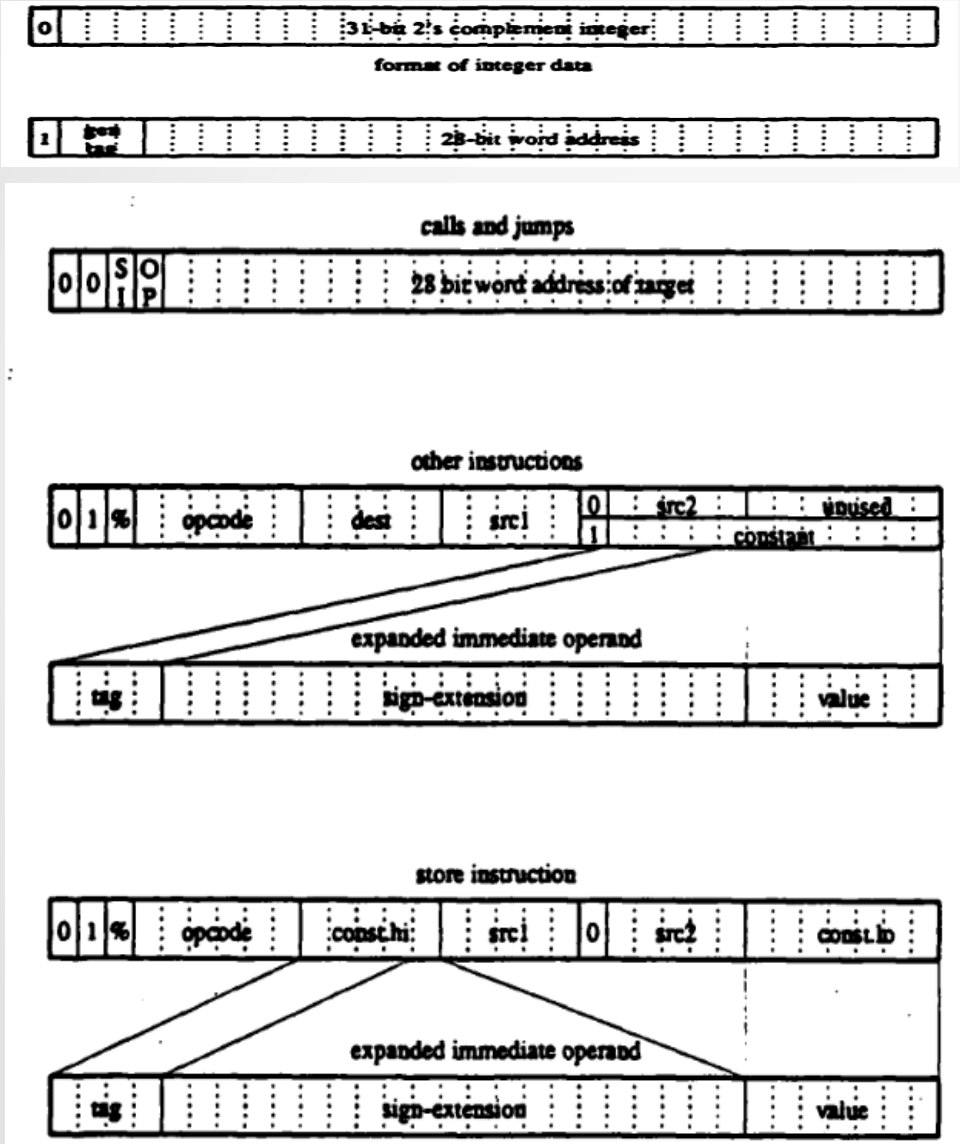
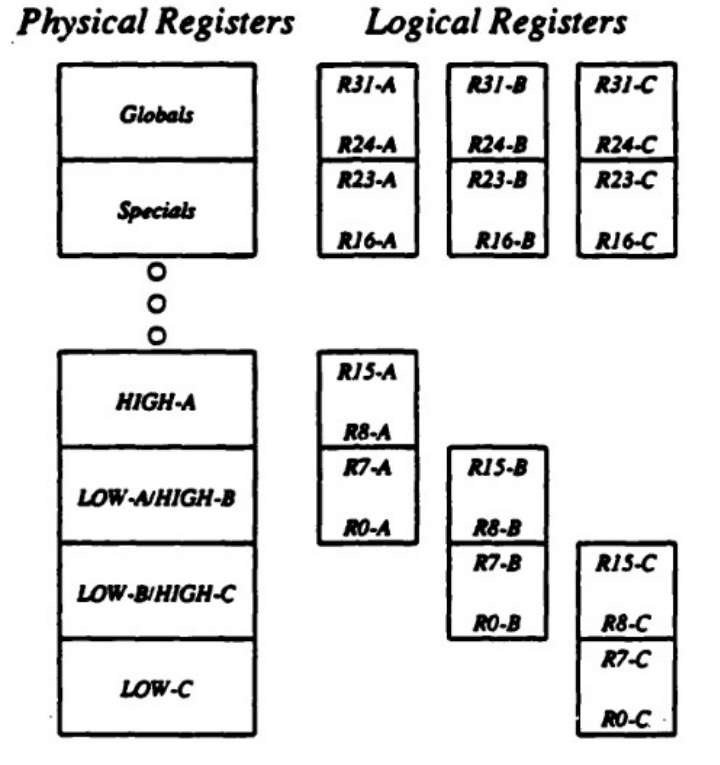
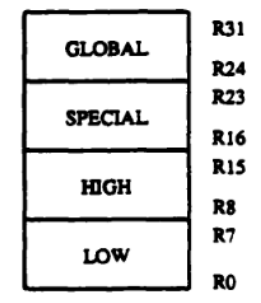


Table 3.2: SOAR Instruction Set.

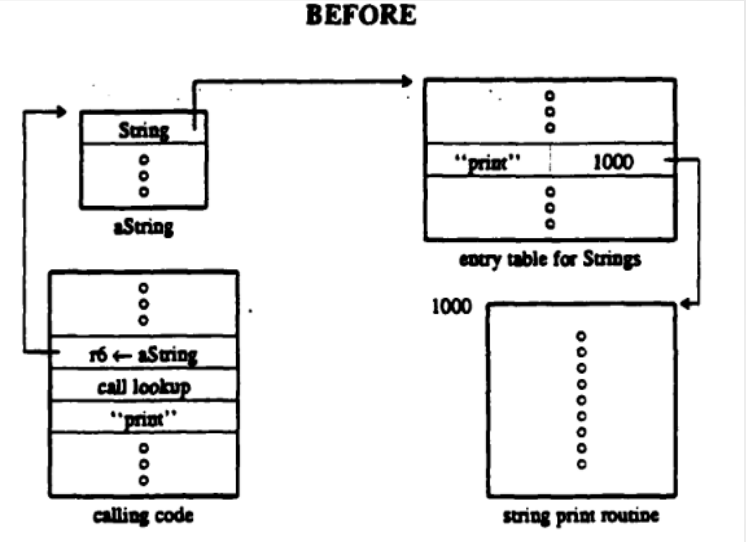
opcode <28:23>$_8$	Instruction	Operands	Cycles	Operation
10–17	[%]ret[w][i][n]	rs, const	2	pc ← rs + const Options as part of return: [%] Disables return address tag checking (non-LIFO a.r.) [w] Change register window [i] Enable Interrupts [n] Initialize r8, ..., r13
50	[%]add	rs, s2, rd	1	rd ← rs + s2
52	[%]sub	rs, s2, rd	1	rd ← rs - s2
44	[%]xor	rs, s2, rd	1	rd ← rs xor s2
46	[%]and	rs, s2, rd	1	rd ← rs & s2
47	[%]or	rs, s2, rd	1	rd ← rs | s2
51	[%]sll†	rs, rd	1	rd ← rs + rs (Left shift)
40	[%]srl	rs, rd	1	rd ← rs shift right logical 1 bit
42	[%]sra	rs, rd	1	rd ← rs shift right arithmetic 1 bit
56	[%]insert	rs, s2, rd	1	rd ← 0; byte s2<1:0> of rd ← rs<7:0>
54	[%]extract	rs, s2, rd	1	rd<7:0> ← byte s2<1:0> of rs: rd<31:8> ← 0
34	[%]load	(rs)s2, rd	2	rd ← M[rs + s2]
35	loadc†	(rs)s2, rd	2	rd ← M[rs + s2]
36	%loadm	(rs)s2, rd	2–9	t ← rs – s2, x ← d: Repeat R[x] ← M[t]; x ← x – 1; t ← t – s2; until x < 0.
30	[%]store	rs2, (rs)const	2	M[rs + const] ← rs2
32	%storem	rs2, (rs)const	2–9	t ← rs – const, x ← s2; Repeat M[t] ← R[x]; x ← x – 1; t ← t – const; until x < 0.
20	[%]skip	cond rs, s2	2	if cond(rs, s2) pc ← pc + 2
21–27	[%]trap	cond rs, s2	1–3	if cond(rs, s2) r7 ← pc, pc ← Trap
04	nop			do nothing
05	(internal trap)			see [Pen85b]
06	(internal skip)			see [Pen85b]
60–67	(internal loadi)			see [Pen85b]
70–77	(internal storei)			see [Pen85b]
00–37	[%]call	addr	1	r7 ← pc; pc ← addr, cwp ← cwp – 1
40–77	[%]jump	addr	1	pc ← addr

Hardware: RISC III (SOAR)

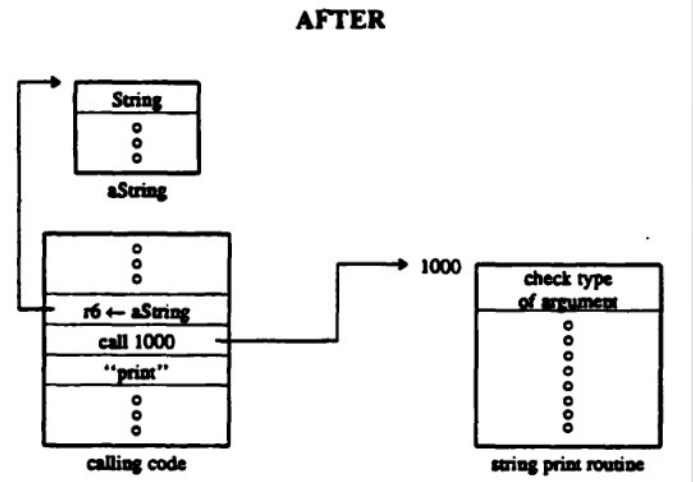




Inline caches



L Peter Deutsch and Alan Schiffman



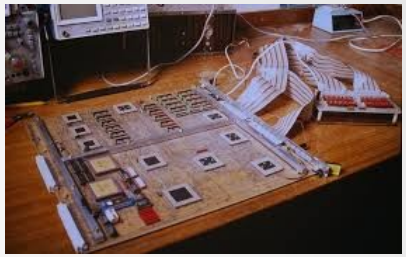
Hardware

COM

1984
Caltech
James T. Kajiya and William J. Dally

Mushroom

1987
University of Manchester
Ifor Wyn Williams, Mario I. Wolczko, and Trevor P. Hopkins

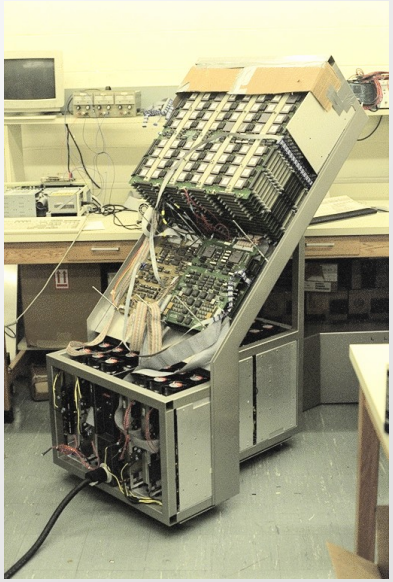


Rekursive

1986
Linn
David Harland

J-Machine

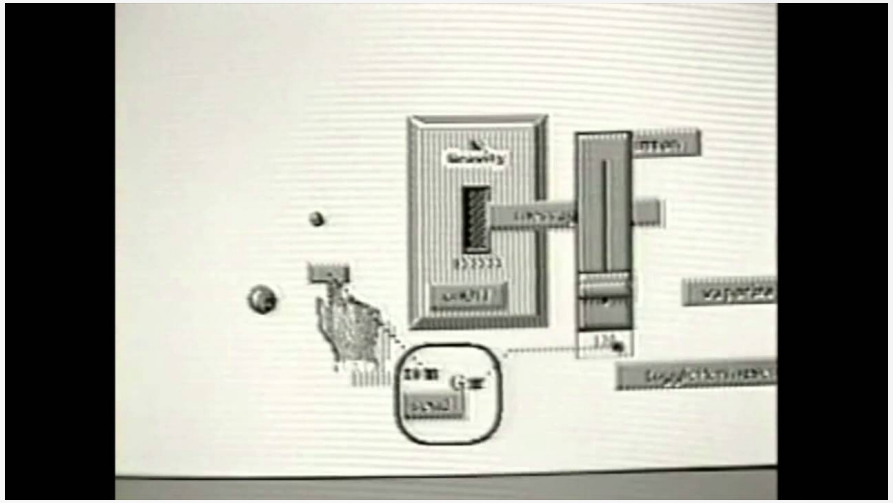
1988
MIT
William J. Dally and others



Self: The Power of Simplicity

Smalltalk-86 call for projects at PARC

David Ungar and his graduate students at Stanford, later Sun



ARK: Alternate Reality Kit by Randy Smith

Self: like Smalltalk only more so

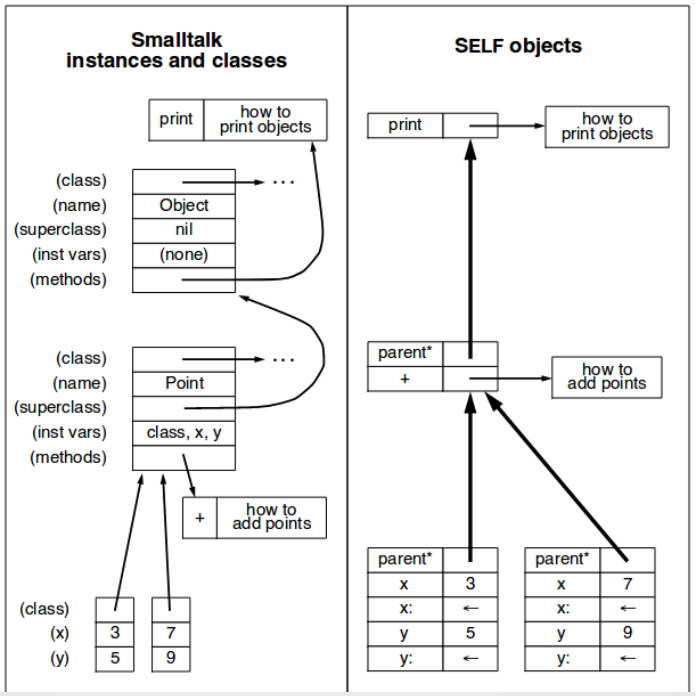
Smalltalk-80

vs

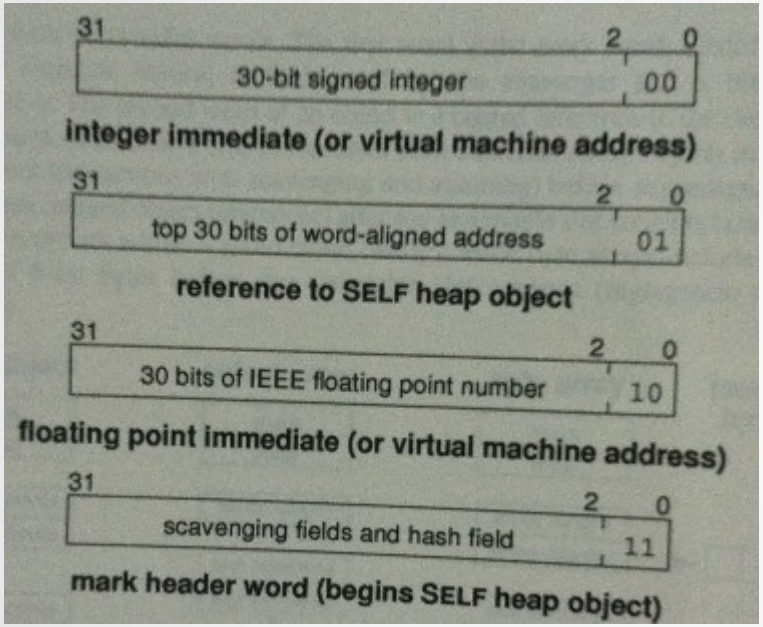
Self

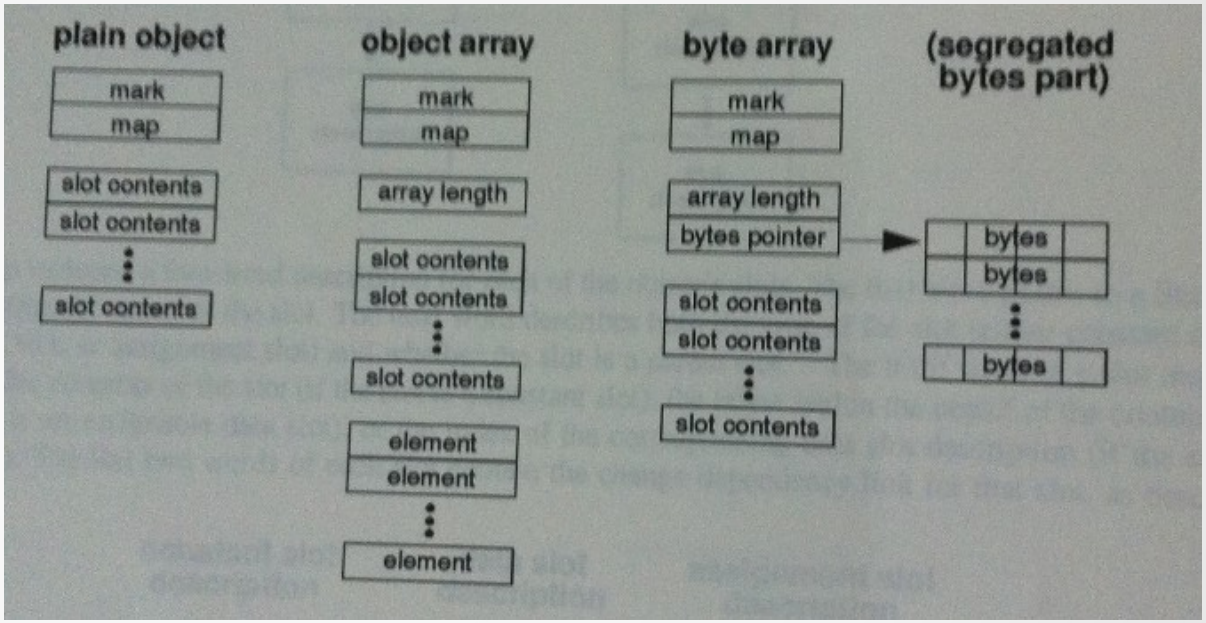
Blocks
Messages
Assignment
Instance variables
Instances
Classes
Metaclasses
Temporary variables
Class variables
Class instance variables
Globals
Pool variables

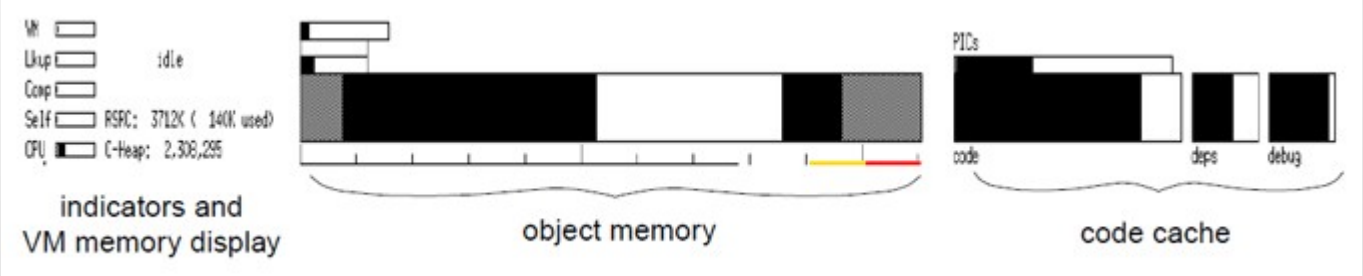
Objects
Named slots (3 kinds)
Parent slots
Blocks
Messages



Self: memory organization







Self 1 and 2: type inference

Craig Chambers

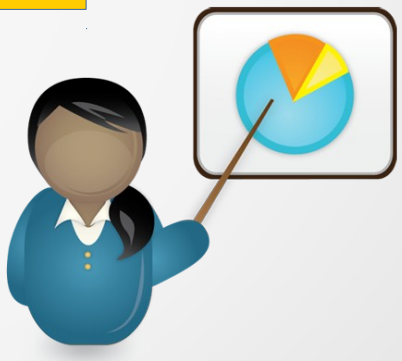
Polymorphism	PICs
Factored code	Inlining
Generic code	Customization
Dynamic types	Type inference

Language evolution

Compiler technology

The computer wants to run

Fortran (or C)

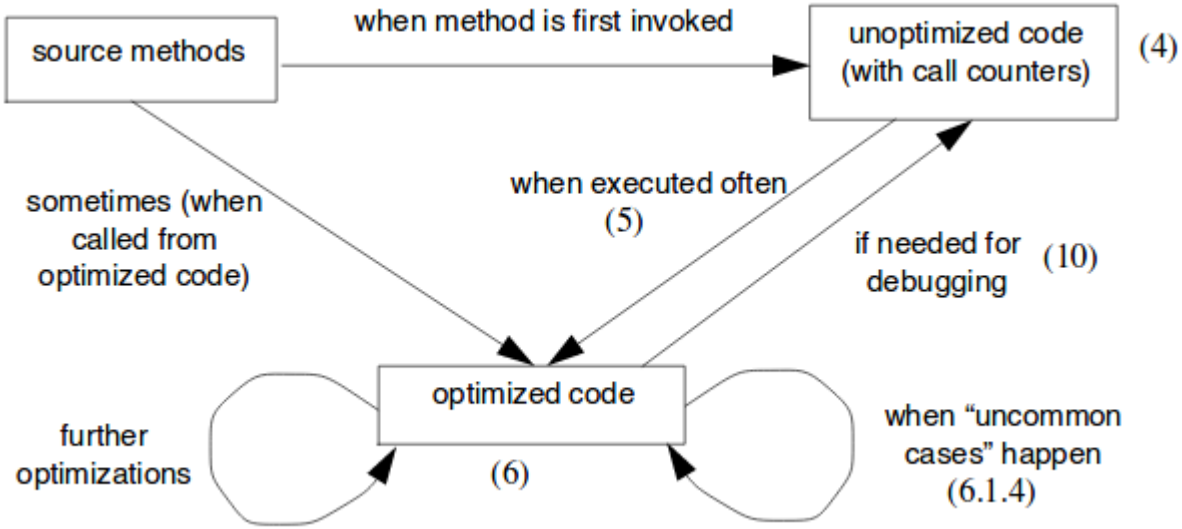


The programmer wants to write in

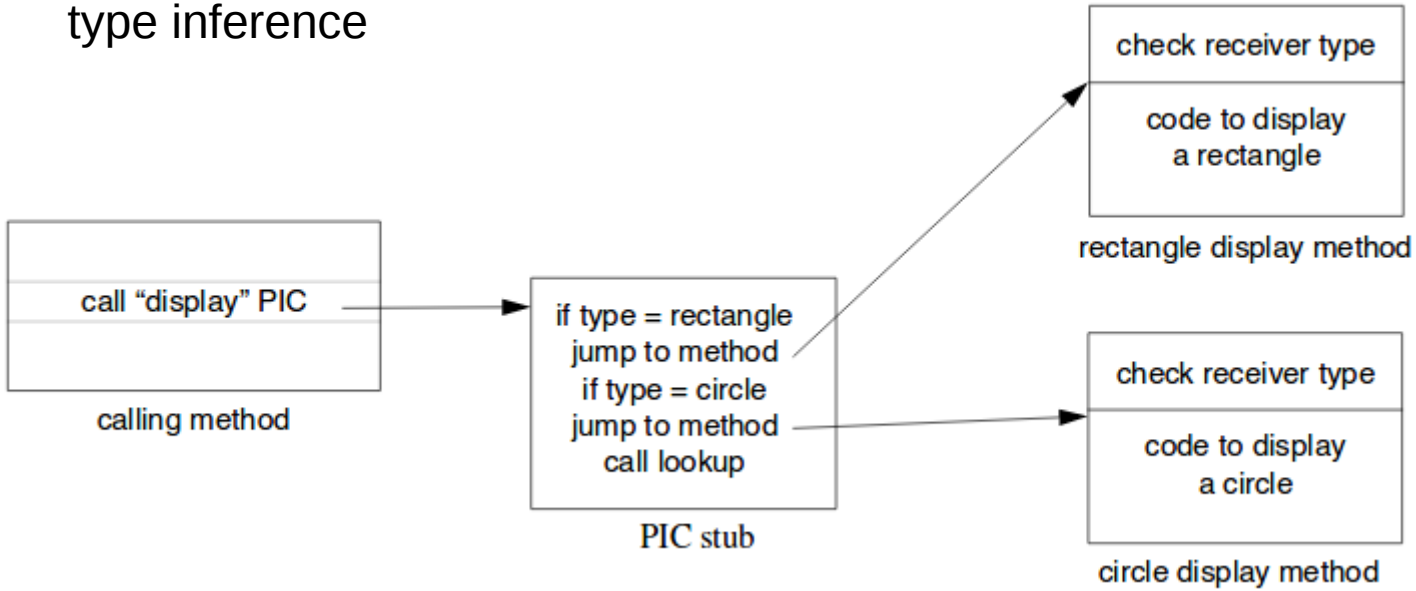
Self

Self 3 and 4: adaptive compilation

Urs Hölzle



Type feedback instead of type inference



Two compilers: simple and fast generates instrumented code, good compiler for "hot spots"

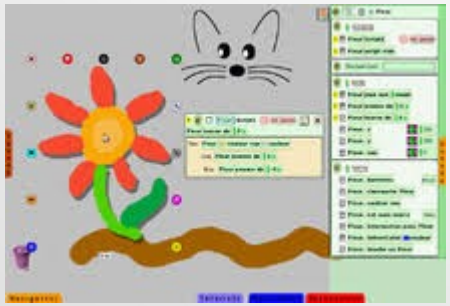
Squeak: Back to the Future

Boxer

Hypercard

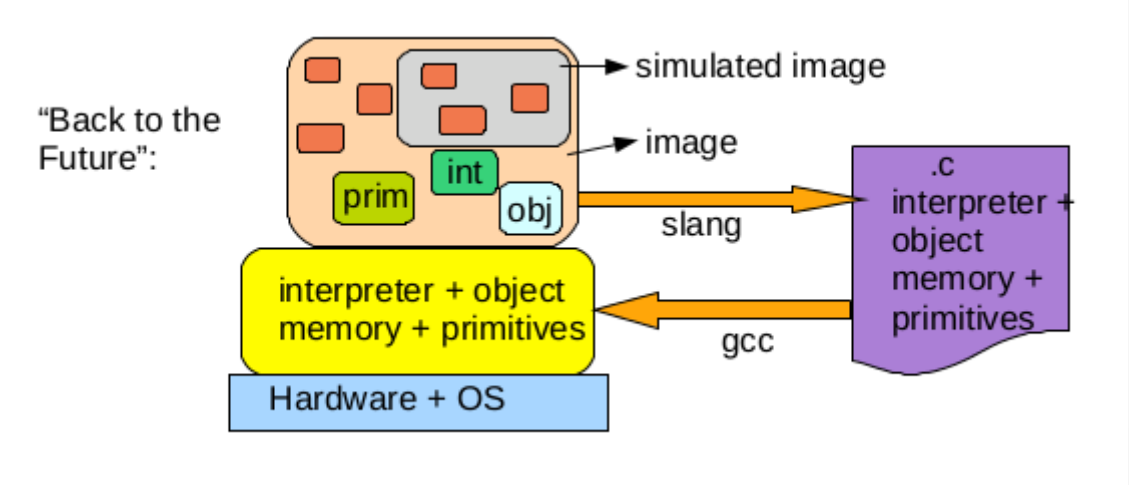
StarLogo

Vivarium



Etoys

Squeak: Slang and Plugins



Squeak: compilers

Ian Piumarta

Jitter 1: threaded code

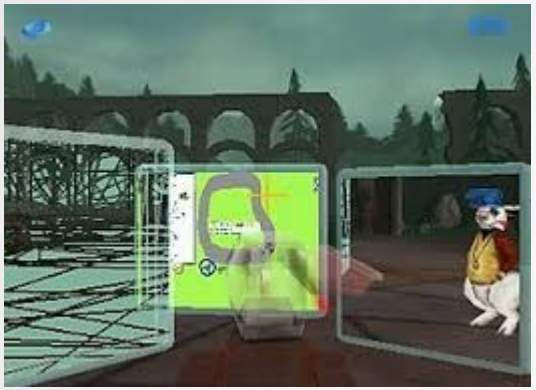
Jitter 2: x86 and PowerPC was released with Squeak 2.0

Jitter 3: interesting experiments

Bryce Kampjes

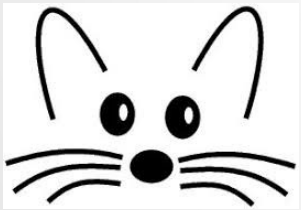
Exupery: bytecode to x86 compiler in the image

Cog => OpenSmalltalk VM



OpenCroquet

Qwaq (later Terf)



Squeak

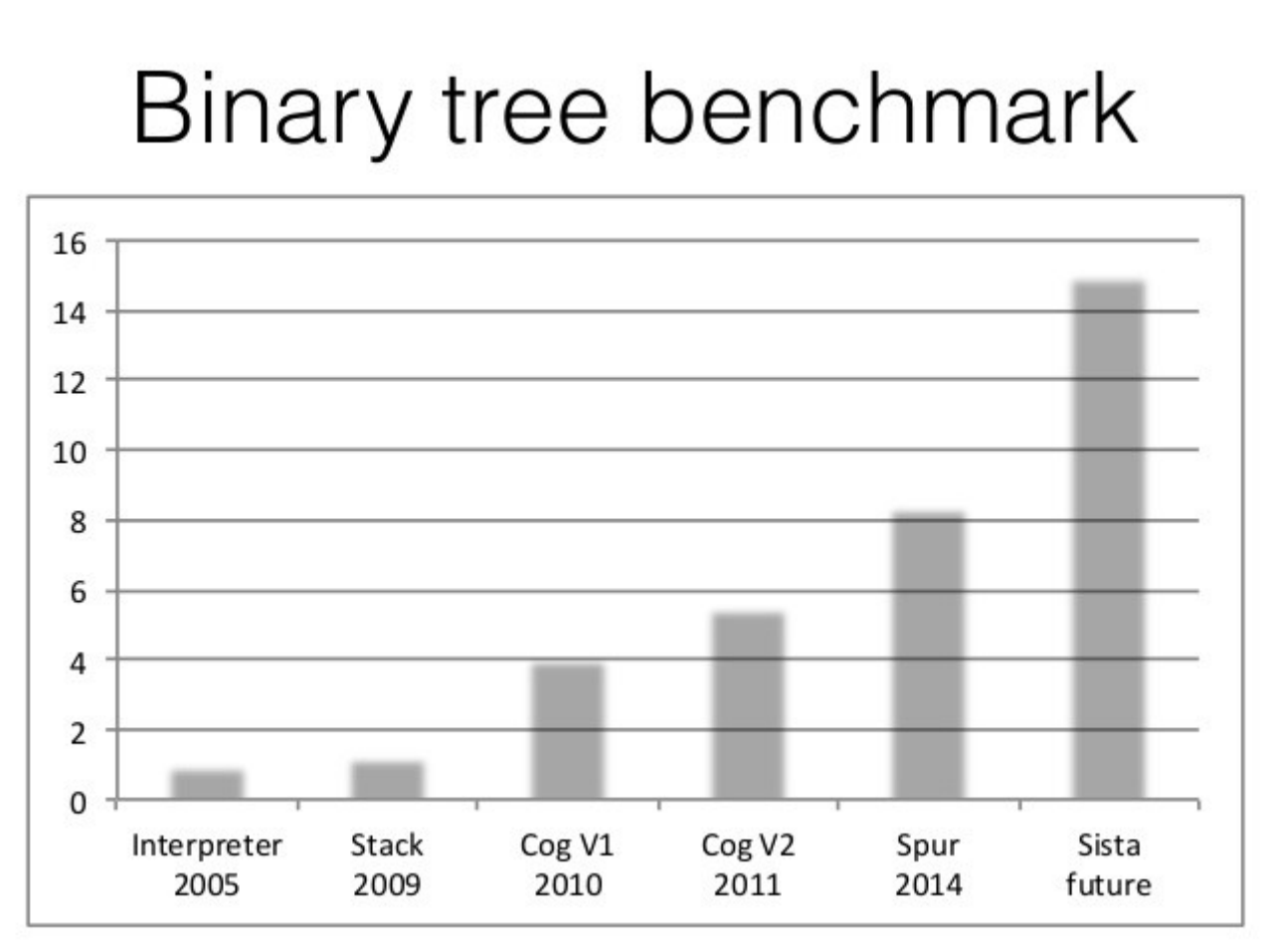
Pharo

Cuis

Newspeak

Eliot Miranda

OpenSmalltalk VM evolution



OpenSmalltalk: object memory

Version 3 → Spur

Binary representation of object pointers ; x is 1 or 0

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0

First bits are 00; this is a direct pointer to an object in the heap

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1

First bit is 1; this is a SmallInteger instance (31 bits signed int)

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0

unused tag

Binary representation of object pointers ; x is 1 or 0

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0

First bits are 00; this is a direct pointer to an object in the heap

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1

First bit is 1; this is a SmallInteger instance (31 bits signed int)

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0

First bits are 10; this is a Character instance

Binary representation of object pointers ; x is 1 or 0

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	

First bits are 000; this is a direct pointer to an object in the heap

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1	

First bit are 001; this is a SmallInteger instance (61 bits signed int)

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0	

First bits are 010; this is a Character instance

x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	

First bits are 100; this is a SmallFloat instance

Spur's object header

s	s	s	s	s	s	s	s	x	x	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h	h
x	x	x	f	f	f	f	f	x	x	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c

s number of slots f object format x remaining bits

h identity hash c class index

References

- “The Early History of Smalltalk”, Alan Kay, History of programming languages---II, pages 511-598, 1993
- “The Design and Implementation of the Self Compiler, an Optimizing Compiler for Object-Oriented Programming Languages”, Craig Chambers, PhD Thesis, Stanford University, 1992
- “Adaptive optimization for Self: Reconciling High Performance with Exploratory Programming”, Urs Hölzle, PhD Thesis, Stanford University, 1994
- Cog Blog, Eliot Miranda