Object-Oriented Extensions

Jecel Mattos de Assumpção Jr
jecel@merlintec.com

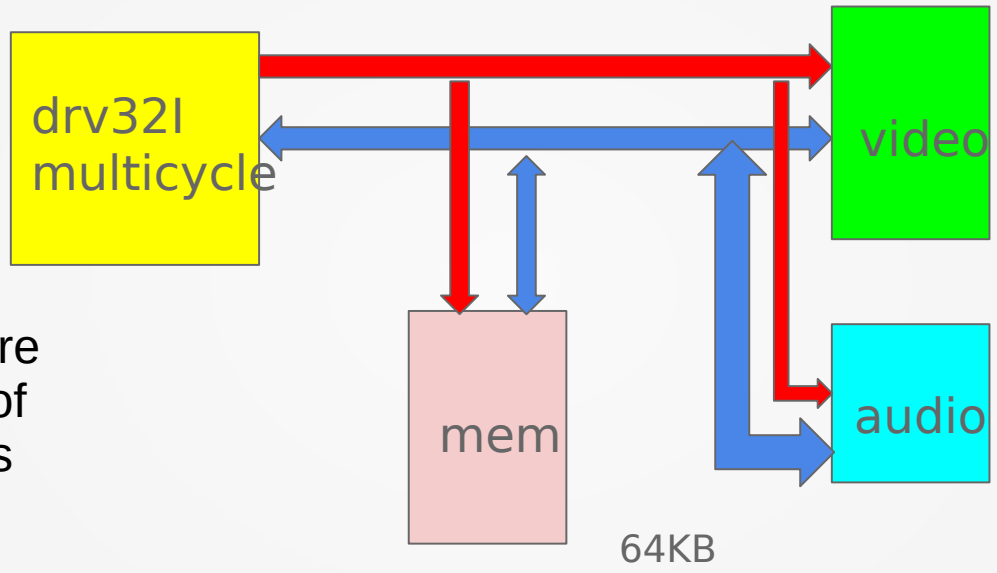
RISC-V J Extension Meeting
July 18, 2024

Motivation

Book introducing computer architecture through the history of videogame consoles

- Portuguese
- English
- Spanish
- French

Third generation

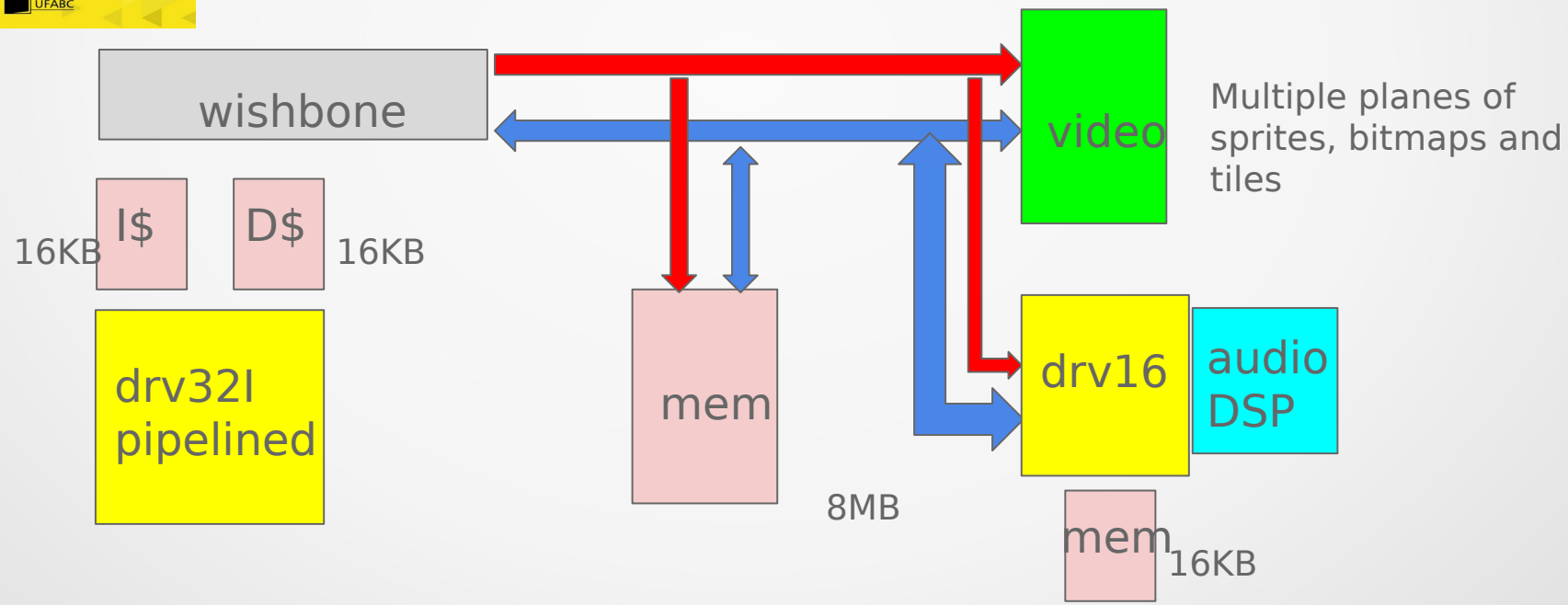


40x30 text with
h&v scroll
32 sprites
2 bits per pixel

3 voices:
Pulse, Triangle,
Sawtooth,
Noise
FM:
26 operators

Motivation

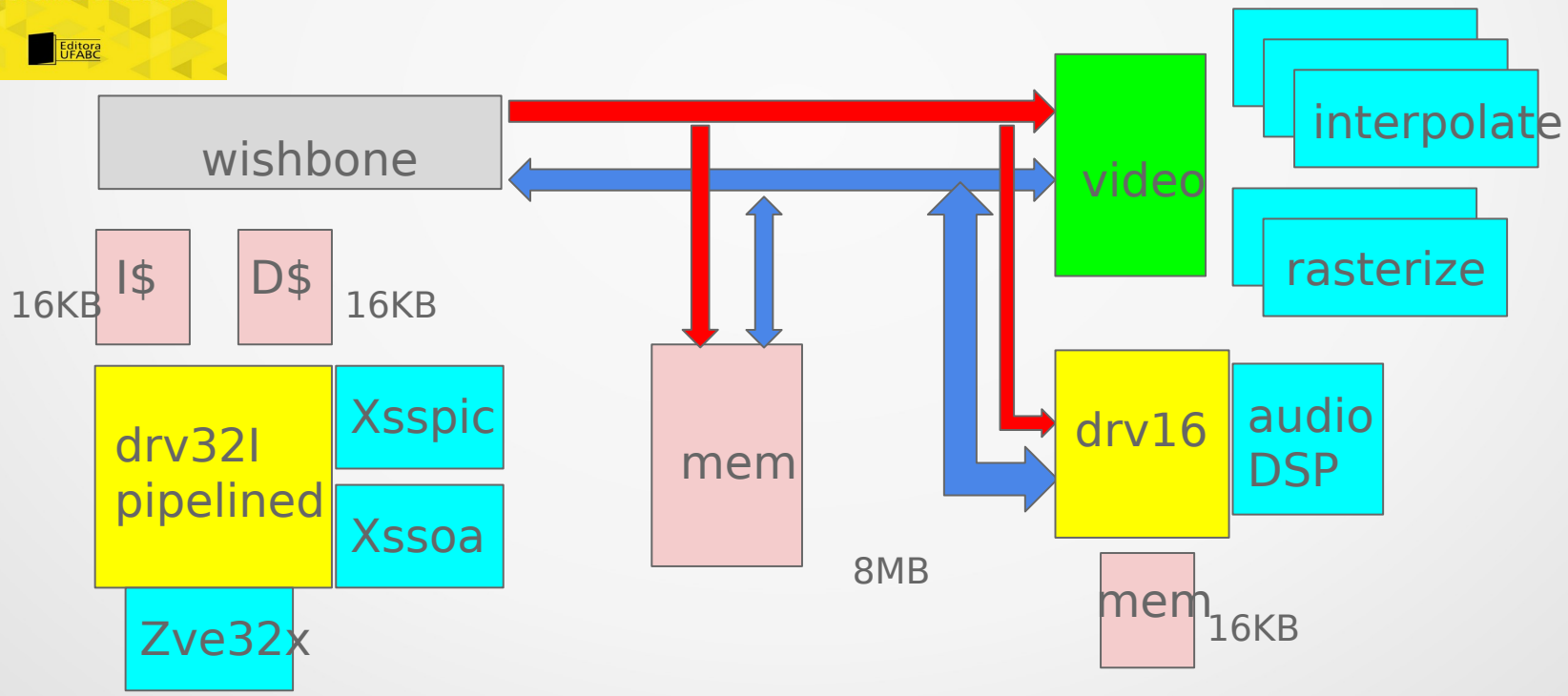
Fourth generation



Motivation

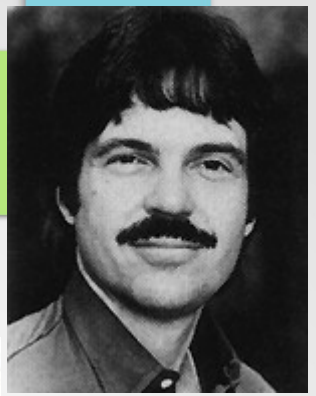
Idea: combine book with “children’s computer” project (Neo Smalltalk)

Fifth generation

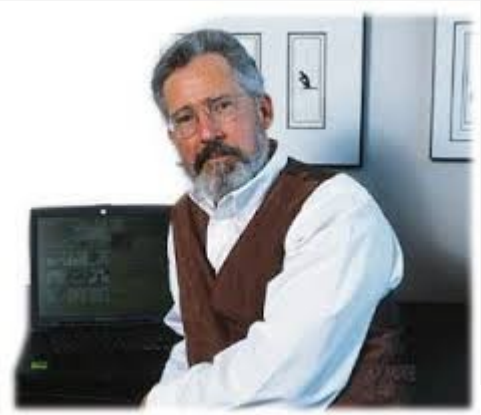


Smalltalk-72: one page design

Alan Kay



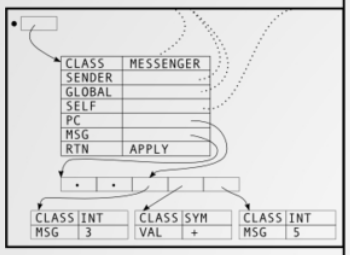
Dan Ingalls



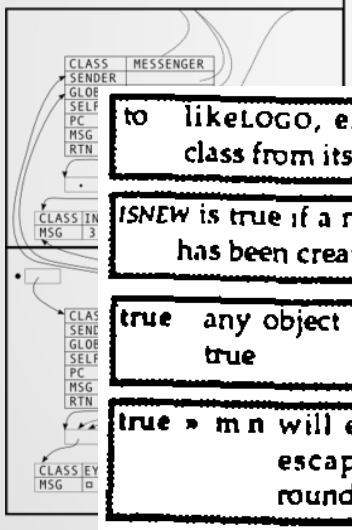
DG Nova Basic => Nova assembly

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 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 va
    (ISNEW  » (:h. :t)              "cons—if no explicit return is given, se
    □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

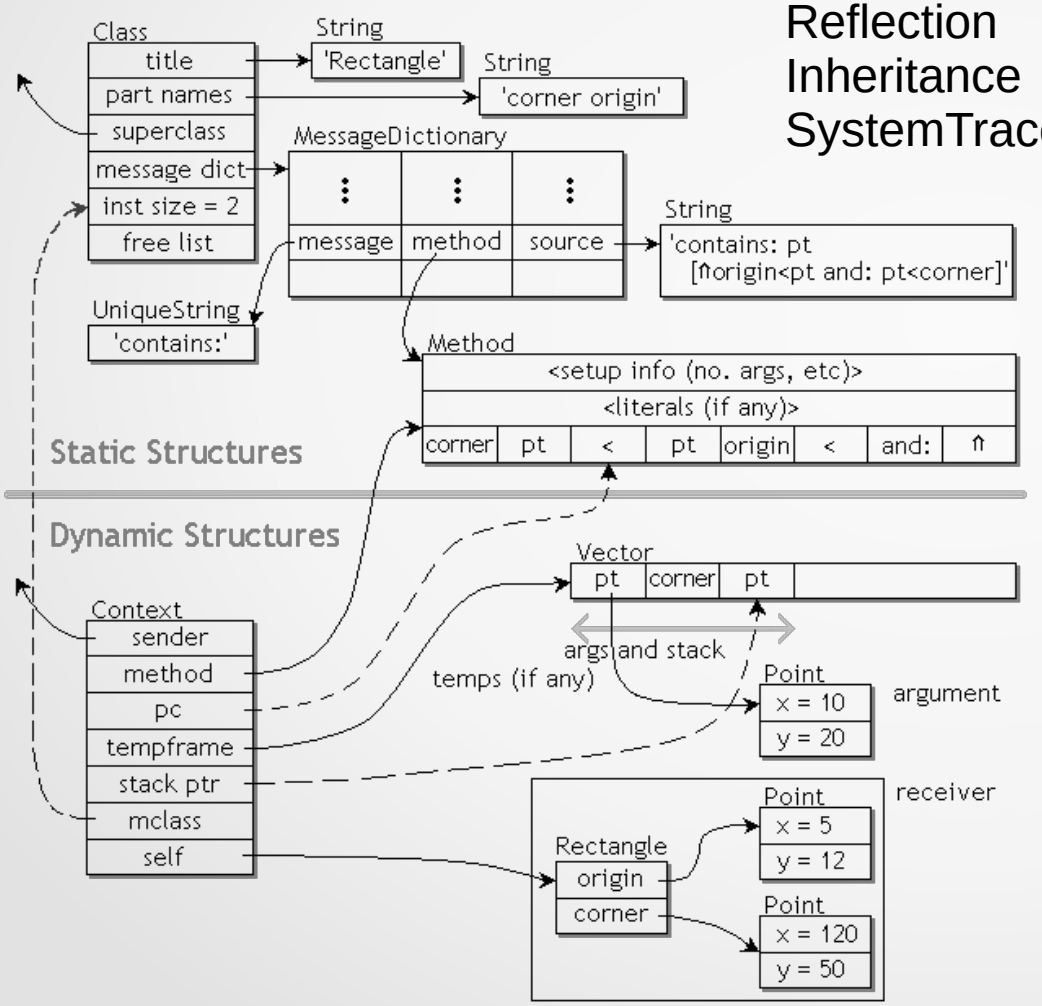
^ send-back returns its value to sender

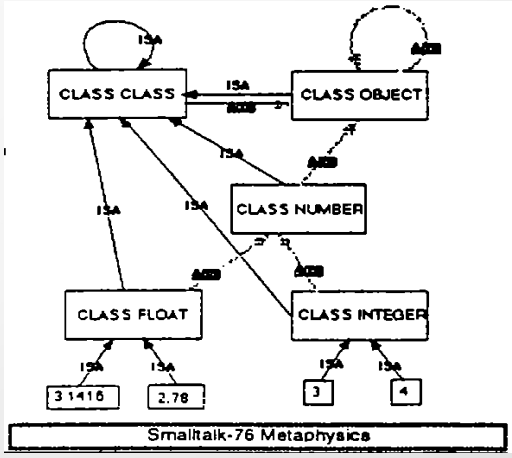
. "statement separator" value is following message

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

Smalltalk-76

Fixed syntax
Bytecodes
Reflection
Inheritance
SystemTracer

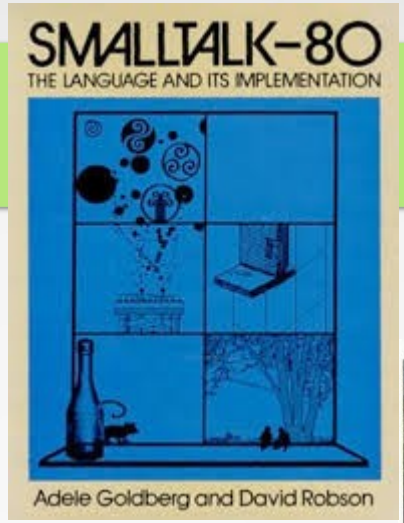




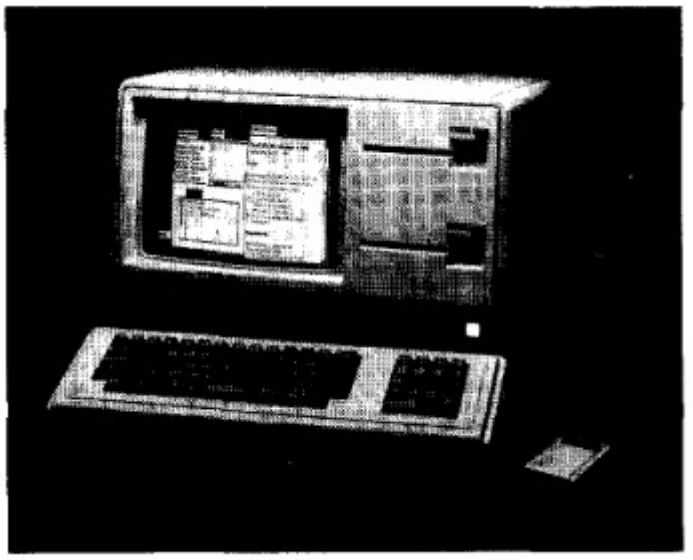
Smalltalk-78 and the Notetaker

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

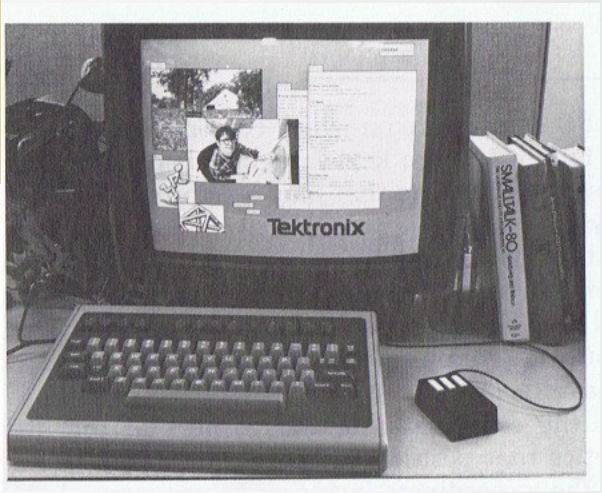
Smalltalk-80



Adele Goldberg

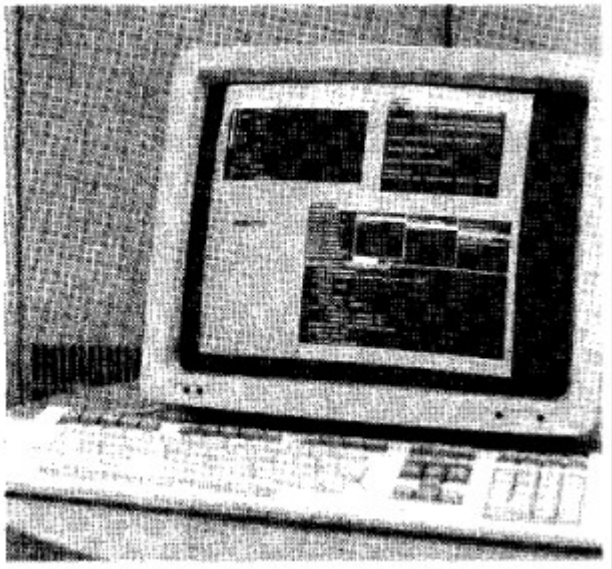


Apple



Tektronix

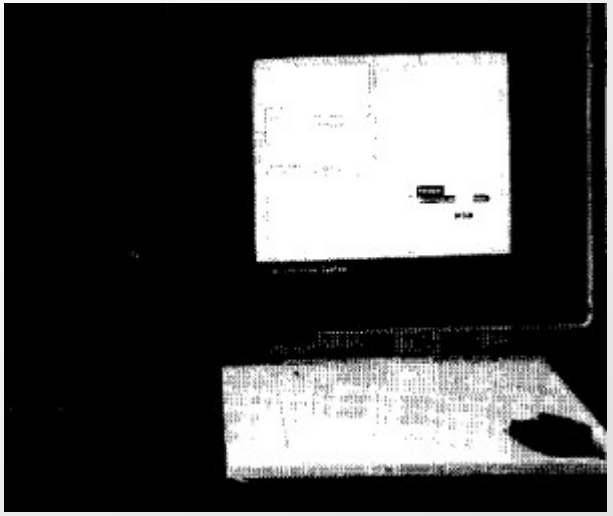
David Patterson's quantative approach



DEC => Berkeley

BS on Sun 1

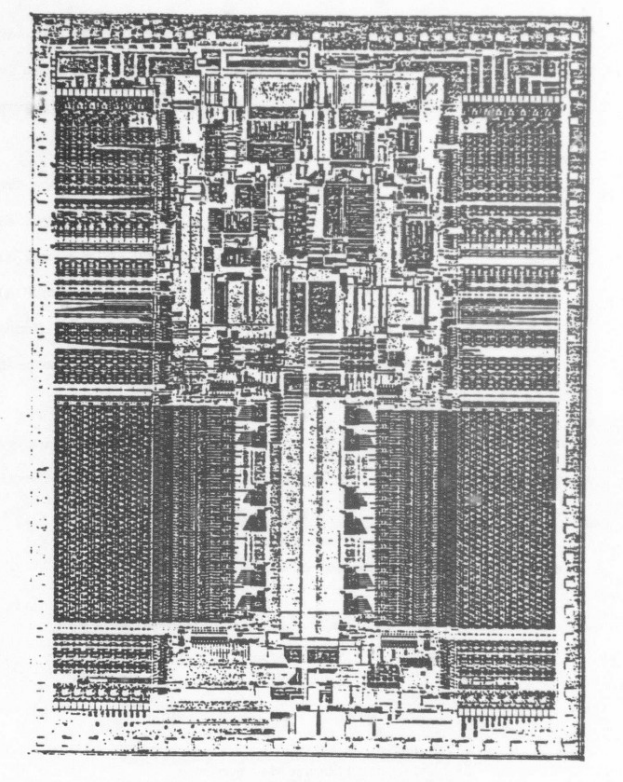
HP



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

1984: Dynamic Compilation (JIT)

L Peter Deutsch

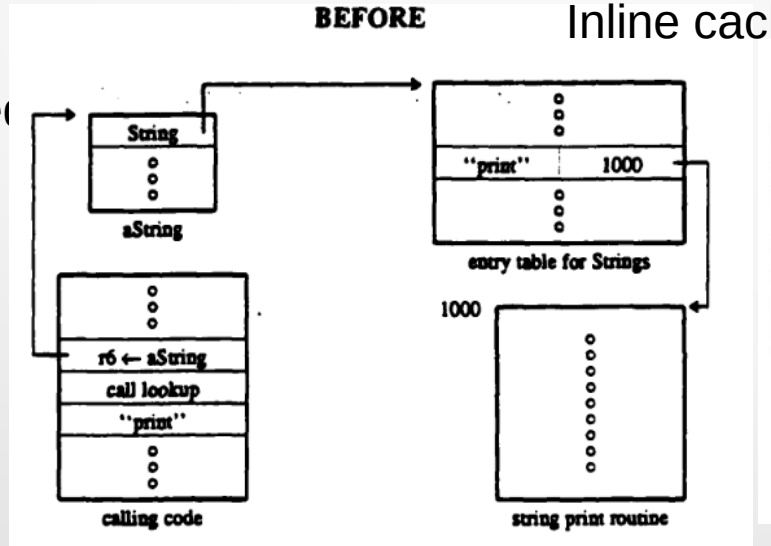
Allan M. Schiffman

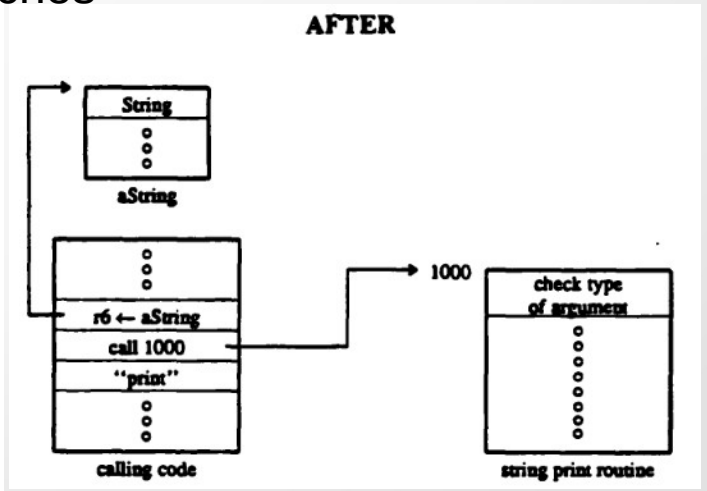
Interpreted Code

Compile

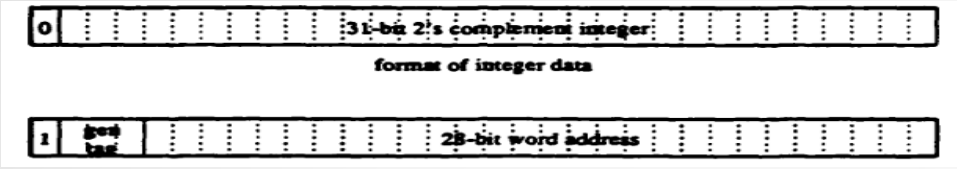
Compiler

Inline caches





Hardware: RISC III (SOAR)



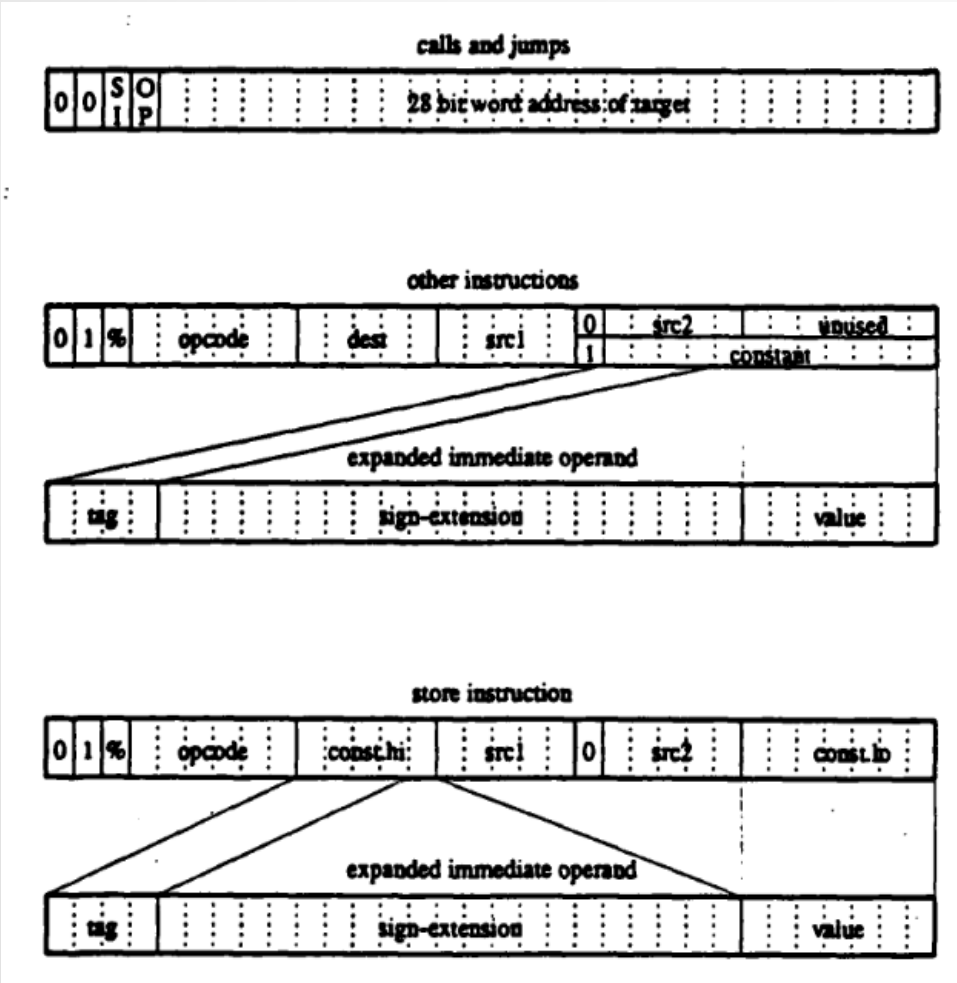


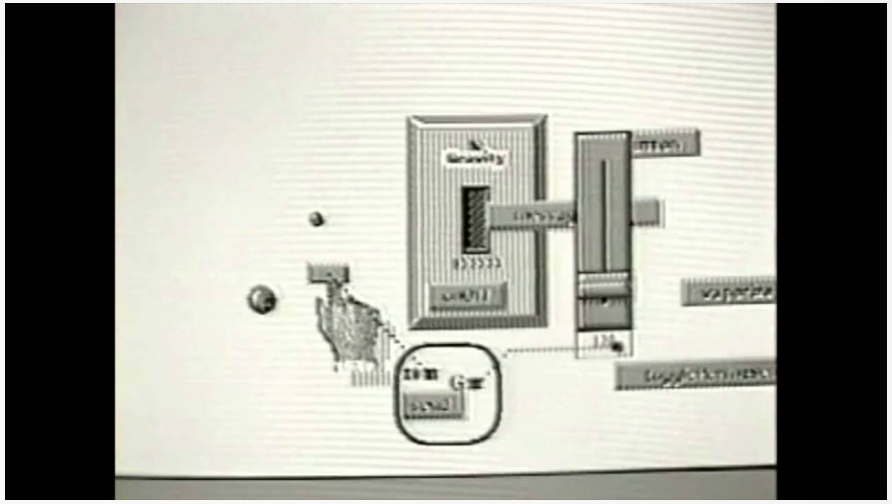
Table 3.2: SOAR Instruction Set.

opcode <28:23>	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

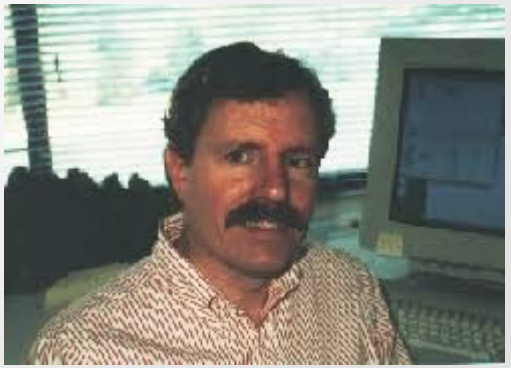
Self: The Power of Simplicity

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

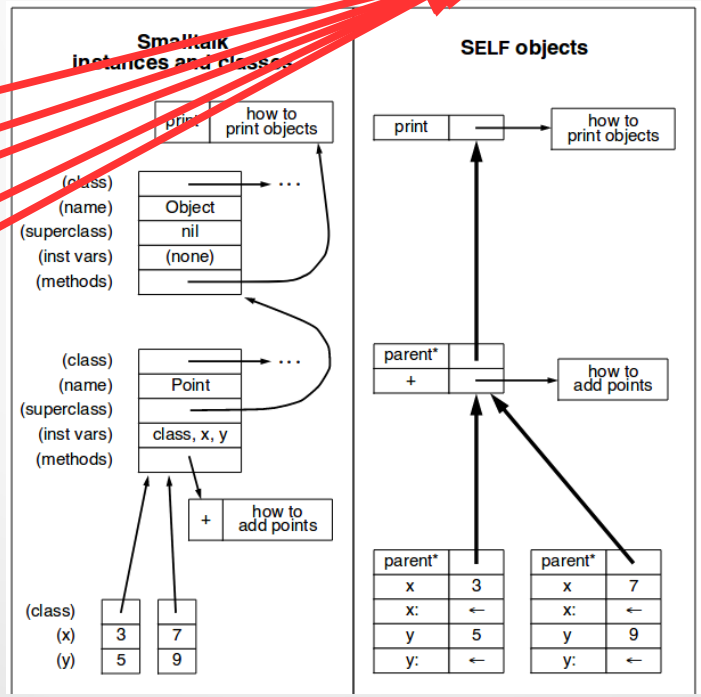
Smalltalk-80

vs

Self

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

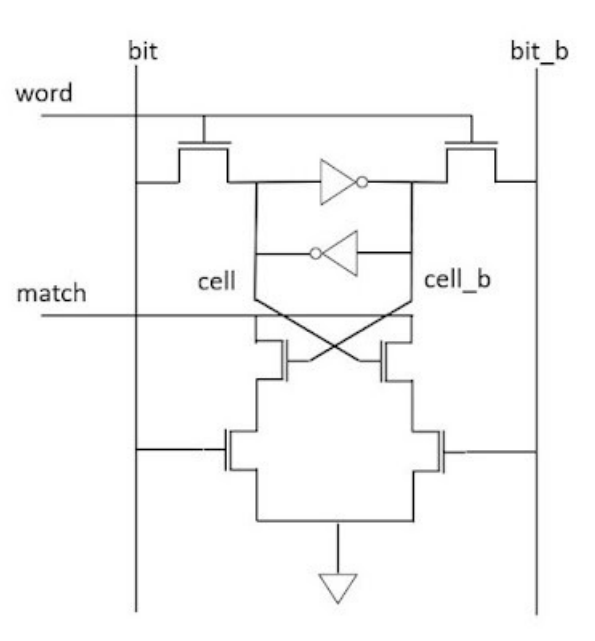
Objects
Named slots (3 kinds)
Parent slots
Blocks
Messages



Some Smalltalk Hardware

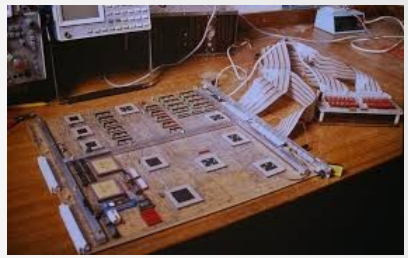
OO Processor

1990
LSI Polytechnical School, University of São Paulo
Jecel Assumpção Jr,



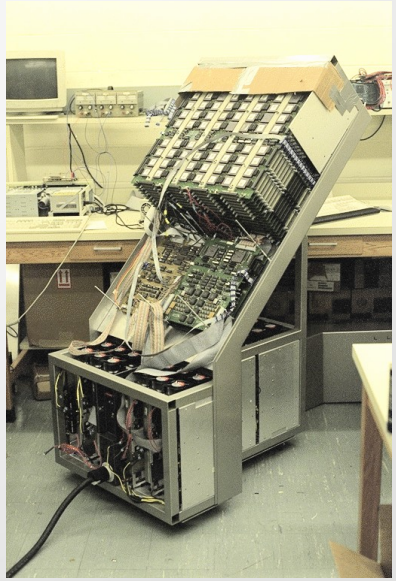
Mushroom

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



J-Machine

1988
MIT
William J. Dally and others

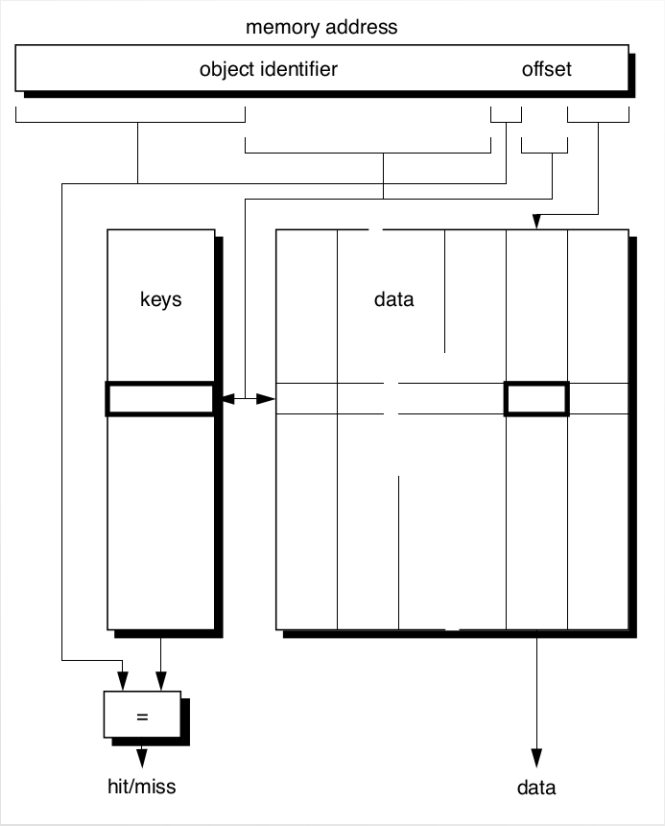


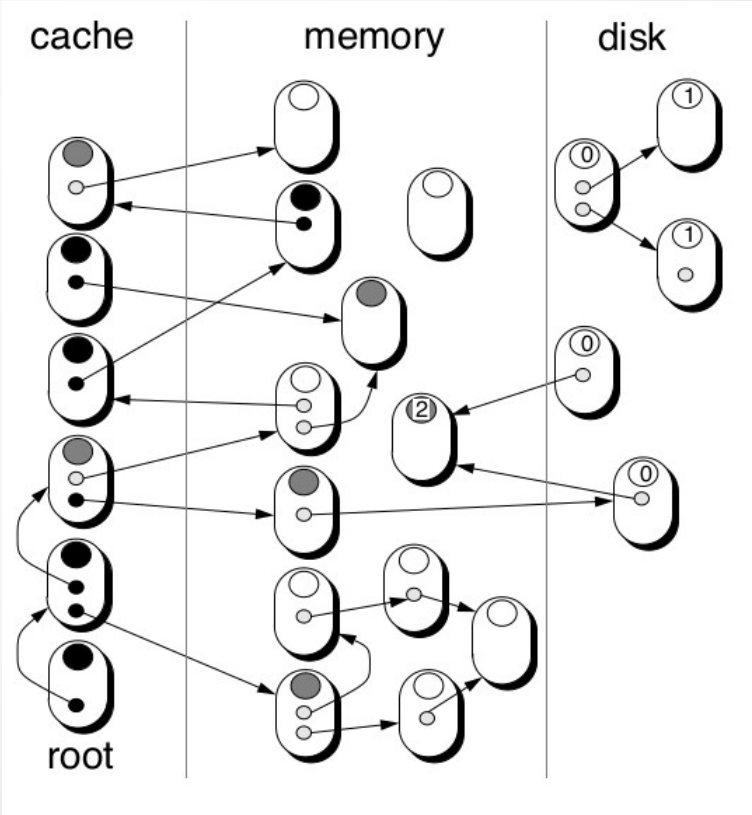
Mushroom: Object Addressing

Trevor Hopkins

Ifor Williams

Mario Wolczko





1990: the evolution of the Self virtual machine

Craig Chambers

Self 1.0

Self 2.0

Self 1 and 2: type inference

Craig Chambers

Polymorphism

Factored code

Generic code

Dynamic types

PICs

Inlining

Customization

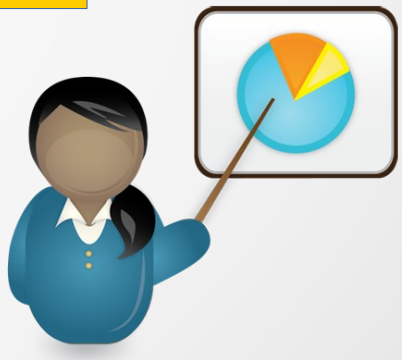
Type inference

Language evolution

Compiler technology

The computer wants to run

Fortran (or C)

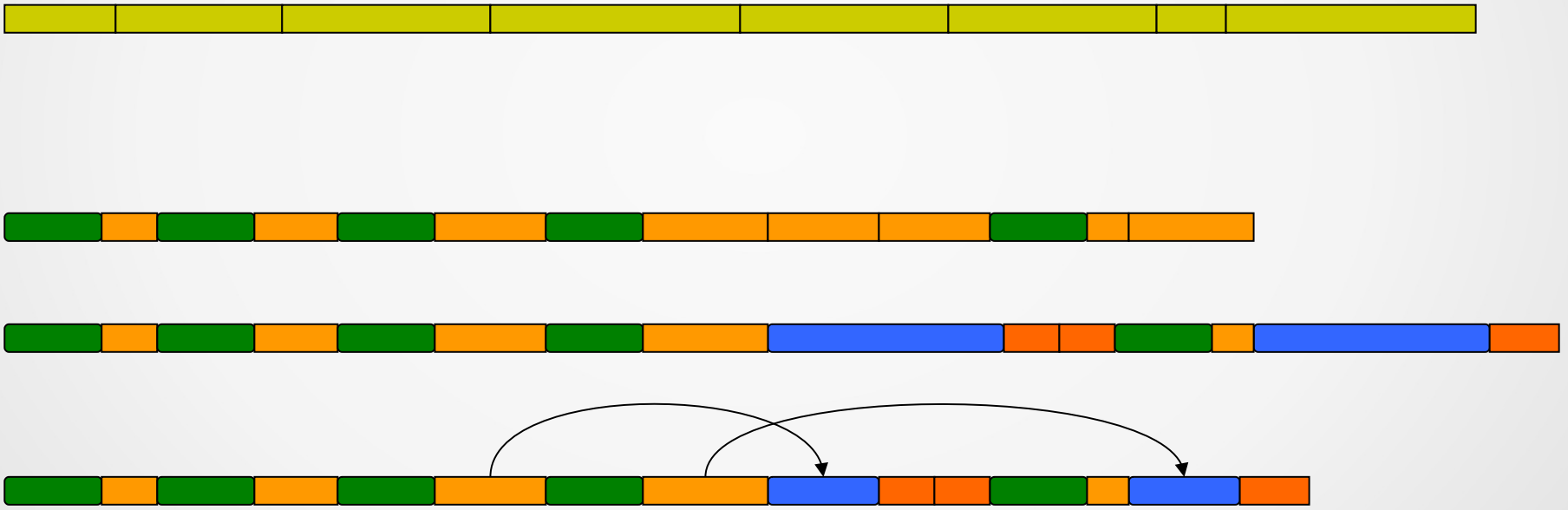


The programmer wants to write in

Self

1992: Adaptive Compilation

Urs Hölzle

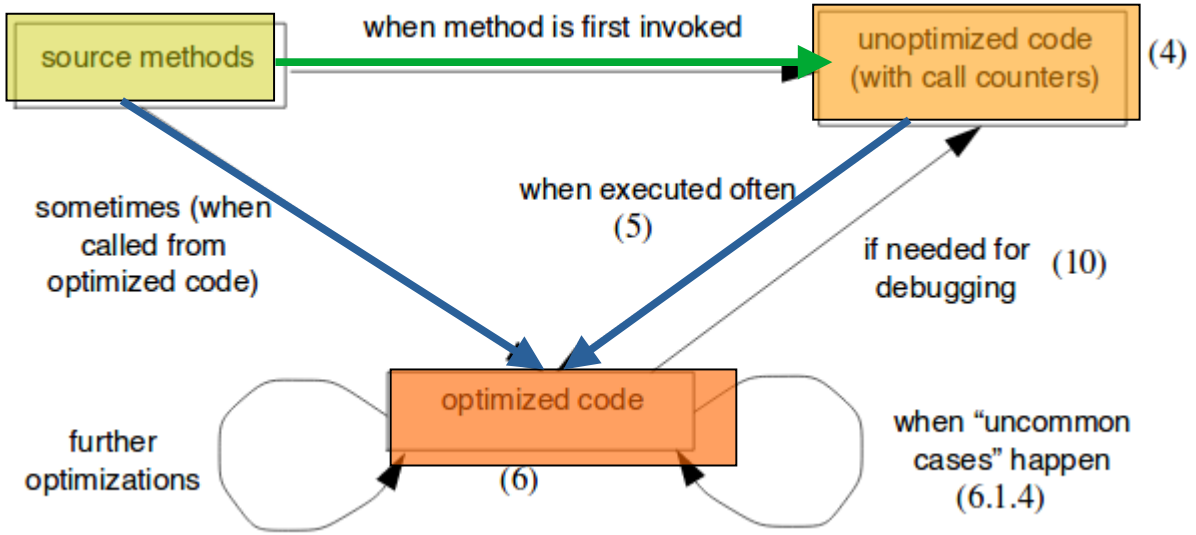


“type feedback”

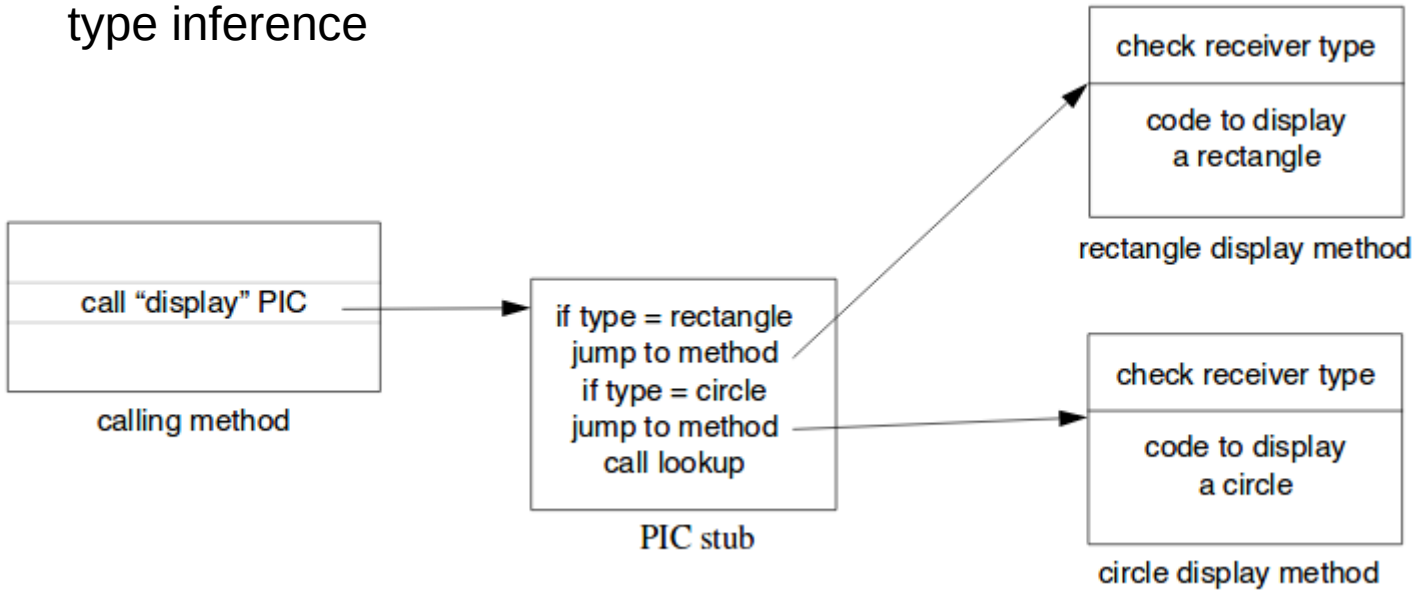
Self 3.0

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"

SiliconSqueak history

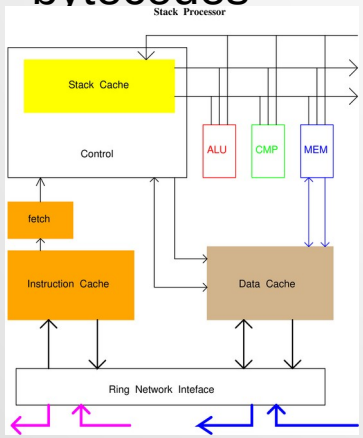
1998: Tachyon – VLIW 4 x 16 bit MOVE

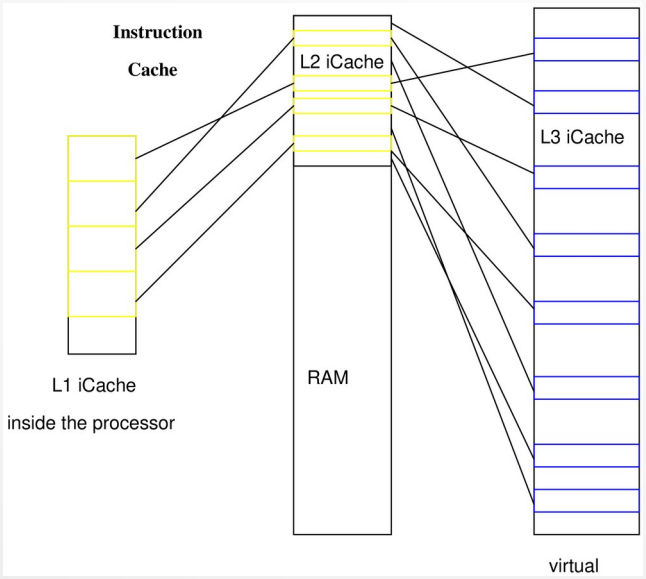
<432,0>	[true]	R6 -> PIC_TRIG	[true]	CONST32 -> PIC_OP	#????			
<IntMap>	[true]	R0-> ADDER_OP	[true]	R2 -> ADDER_TRIG	[true]	#7 -> CR	[...]	..
<FloatMap>	[true]	R0 -> FADD_OP	[true]	R2 -> FADD_TRIG	[true]	#3 -> CR	[...]	..
<XXXXXX>	ZZZZZZZZZ							
....								
<432,3>	[true]	FADD_OUT -> R0	...		...		...	
....								
<432,7>	[true]	ADDER_OUT -> R0	...		...		...	

3 way PIC instruction

PIC size	%
0	12.4%
1	81.9%
2	4.4%
3	0.8%
4	0.2%
5-10	0.2%
11-99	0.05%
>99	0.02%

2001: Plurion – stack-based bytecodes

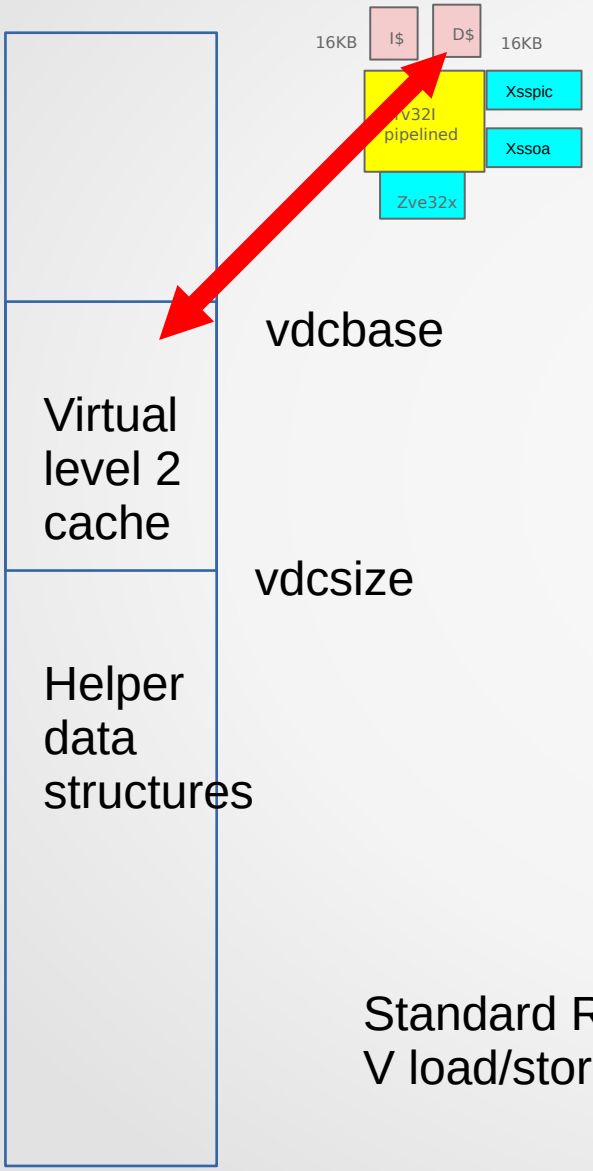




Virtual level 2 caches

2006: RISC42, 2008: SiliconSqueak

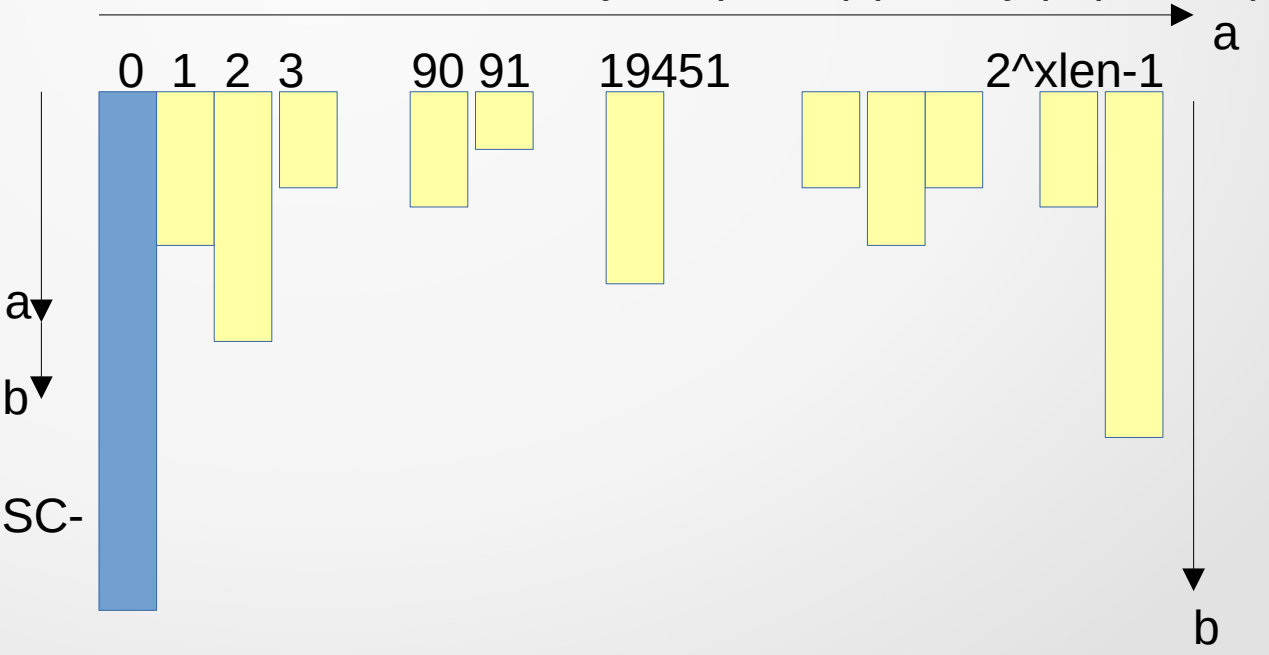
Xssoa: Object Addressing



New Instructions:

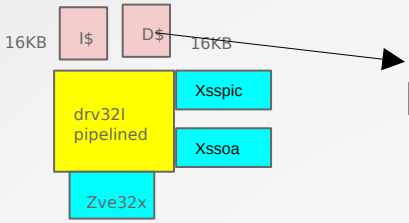
ldod d,b(a)	ldow d,b(a)	ldowu d,b(a)	ldoh d,b(a)
ldohu d,b(a)	ldob d,b(a)	ldobu a,b(a)	ldodi d,b(a)
ldowi d,b(a)	ldowui d,b(a)	ldohi d,b(a)	ldohui d,b(a)
ldobi d,b(a)	ldobui d,b(a)	stod c,b(a)	stow c,b(a)
stoh c,b(a)	stob c,b(a)	stodi c,b(a)	stowi c,b(a)
stohi c,b(a)	stobi c,b(a)		

2D addressed virtual object space (sparsely populated)



Standard RISC-V load/store

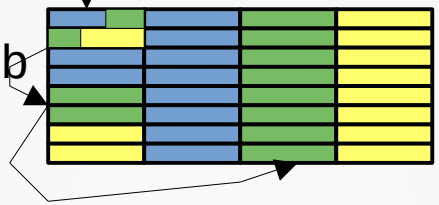
Xssoa: virtual level 2 cache



BlockNum = Hash1(a,b) % (vdcsize/BlockSize)

= Hash2(a,b)

= a,b

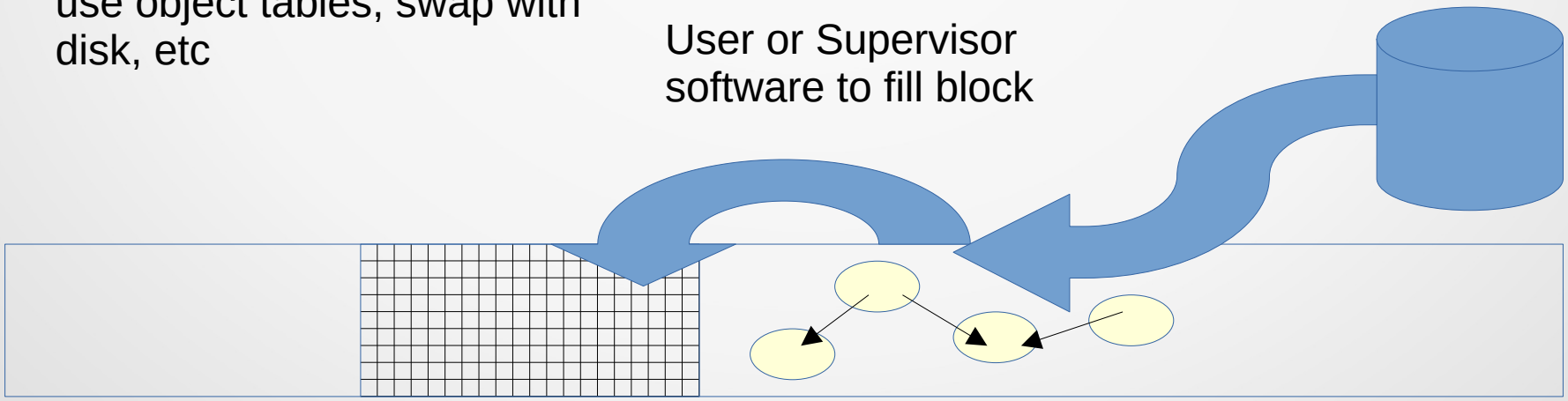


3 way

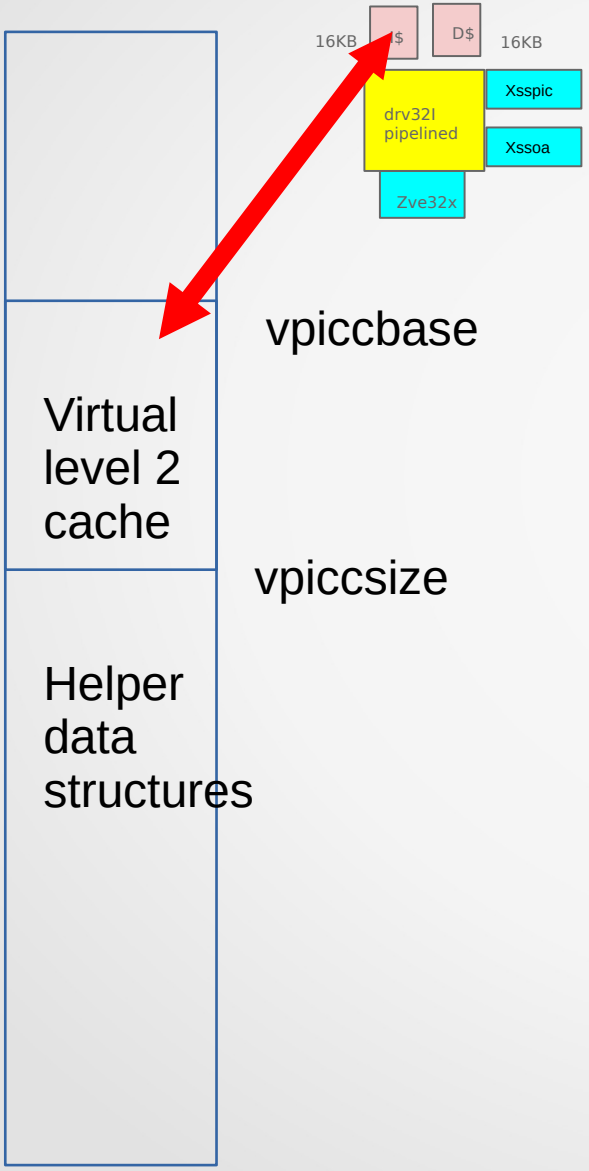
128 bytes RV32, 3 way
256 bytes RV64, 3 way
512 bytes RV32, 7 way
1024 bytes RV64, 7 way

Can: compress/decompress, use object tables, swap with disk, etc

User or Supervisor software to fill block



Xsspic: PIC Mode

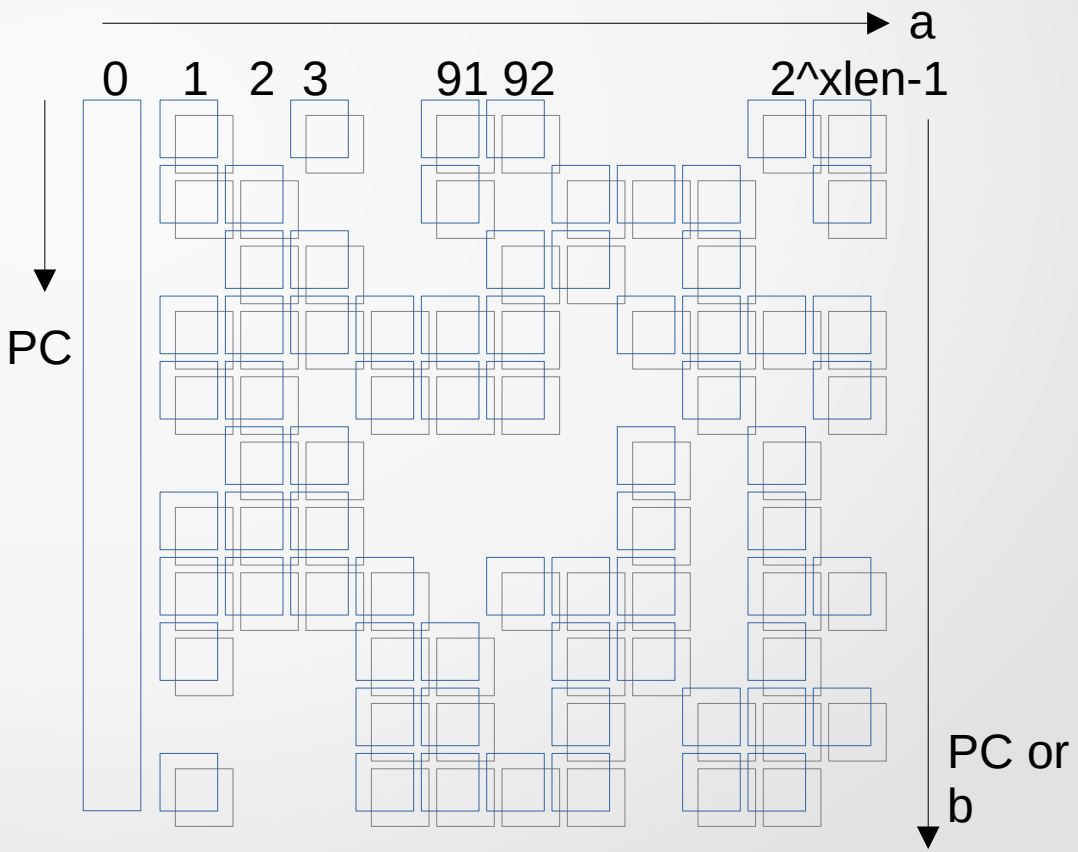


New instructions:

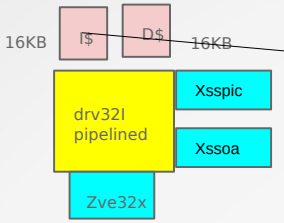
pic a

pici a,b

2D+ PIC virtual space (sparsely populated)



Xsspic: virtual level 2 cache



BlockNum = Hash1(pc,a) % (vpiccsize/BlockSize)

= Hash2(pc,a)

= pc,a

3 way

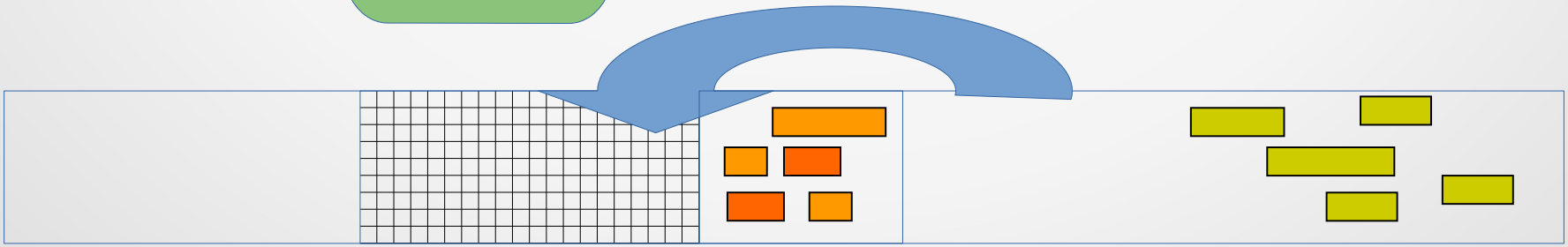
128 bytes RV32, 3 way
256 bytes RV64, 3 way
512 bytes RV32, 7 way
1024 bytes RV64, 7 way

PIC Mode:

```
……
    ldw x6,0(x5)
    pic x6
    add x7,x7,1
…..
```

```
ldw x7,12(x5)
jal zero, 0
????
????
????
????
????
????
```

User or Supervisor
compiler to fill block



Thanks!

Extensions that were not discussed:

- bytecode execution

- tagged math

- tagged addressing

- NoC-based message passing