Smalltalk Computers, Past and Future

Jecel Mattos de Assumpção Jr
November 13, 2019
Why?

Why build Smalltalk computers?

Who and How?

Who built Smalltalk computers in the past and how did they do it?

When?

When will you have a Smalltalk computer?
### Why?

<table>
<thead>
<tr>
<th>conventional</th>
<th>specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word processors, CAD stations, Raspberry Pi, Merlin, Tektronix 440x</td>
<td></td>
</tr>
</tbody>
</table>

#### Algo
- English Electric KDF9
- Burroughs B5000

#### APL, Pascal, Modula-2
- Novix, Harris RTX-2000, MISC MC17, WISC CPU/16, SC32, MuP21, MSL16, Ignite, i21, F21, E16, MARC4, QSP16, TF2216, Steamer16, MicroCore, J1, SC20, F18 GA144

#### Java
- picoJava, aj102, Cjip, Komodo, FemtoJava, ARM Jazelle, JOP, SHAP, MAJIC

#### Lisp
- Symbolics
- Lisp Machine Inc.
- Texas Instruments
- Xerox D Machines

#### Smalltalk
- Xerox D Machines, Katana32, Swamp, AI32, SOAR, COM, Rekursive, Mushroom, J-Machine
Von Neumann bottleneck

Program that accesses 40MB will take 50 seconds.

- Accessing 40MB at 4 MB/s will take 10 seconds.
- Accessing 40MB at 8 MB/s will take 5 seconds.
Cache 4 times faster than main memory with a 95% hit rate.
Smalltalk

0 tinyBenchmarks

Bytecodes / second

Sends / second
Who? 1968: Smalltalk history

- Alan Kay
- Flex 1968/1969
- Seymour Papert
- Dynabook
- B220 tape format
- Simula I
- Ivan Sutherland's Sketchpad
- LISP 1.5
1973: Xerox Alto

Butler Lampson, Chuck Thacker
1978: Notetaker

Multiple 8086 processors => kernels
Small integers are unboxed
Object Table
1979: Dorado
1980: Smalltalk for partner machines

- Apple
- Tektronix
- DEC => Berkeley
- HP
1981
1982: LSI USP

Osvaldo Cristo

João Zuffo  Takeo Kofuji

TG1000 graphics terminal
old HP minicomputer

VersaBUS modules and VersaDOS
1983: SINDE Supermicro

Multiprocessor Unix:
- microkernel
- no interrupts (but timer)
- files on top of virtual memory
- dynamic libraries

VME bus with modified arbitration

Terminals and Printers
1983: Pegasus (Logo Machine)

Jecel & Fábio Cunha
1983: RISC III (SOAR)

Smalltalk On A RISC

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

Professors David Hodges and David Patterson
1983: RISC III (SOAR)

Table 3.2: SOAR Instruction Set

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Instruction</th>
<th>Operands</th>
<th>Cycles</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-17</td>
<td>[%]rgb[w][i][n]</td>
<td>rs, const</td>
<td>2</td>
<td>pc ← rs + const (as part of return)</td>
</tr>
<tr>
<td>0</td>
<td>[?]DISABLE return address tag checking (non-LIFP a.s.)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>[%]xor</td>
<td>rs, ra, rd</td>
<td>1</td>
<td>rd ← rs ⊕ ra</td>
</tr>
<tr>
<td>46</td>
<td>[%]and</td>
<td>rs, ra, rd</td>
<td>1</td>
<td>rd ← rs &amp; ra</td>
</tr>
<tr>
<td>47</td>
<td>[%]or</td>
<td>rs, ra, rd</td>
<td>1</td>
<td>rd ← rs ⊕ ra</td>
</tr>
<tr>
<td>51</td>
<td>[%]call</td>
<td>addr</td>
<td>1</td>
<td>r7 ← pc;</td>
</tr>
<tr>
<td>52</td>
<td>[%]call</td>
<td>addr</td>
<td>1</td>
<td>pc ← addr, pcw ← pcw - 1</td>
</tr>
<tr>
<td>54</td>
<td>[%]jump</td>
<td>addr</td>
<td>1</td>
<td>pc ← addr</td>
</tr>
</tbody>
</table>

Other instructions:

- [%]add
- [%]sub
- [%]xor
- [%]and
- [%]or
- [%]call
- [%]jump

Expanded immediate operand:

- Tag
- Sign-extension
- Value
1983: RISC III (SOAR)

L Peter Deutsch and Alan Schiffman

Inline caches
1984: Tektronix 4404 (AI Workstation)

CPU: 68010@10 MHz + NS32081 FPU
RAM: 1 – 4 MB
Resolution: 640 x 480
CRT: 13"
HDD: 45 MB
FDD: 320 kB
Interfaces: RS232C, LAN, SCSI, centronics
OS: Unix-like OS
Initial price: $14950
1984: Sword32

Sword32 (Katana32)
bytecodes
ASIC
University of Tokyo
1984: Caltech Object Machine (COM)

Caltech
James T. Kajiya and
William J. Dally
1984: Clone/Reduce

Run-time structures cloned from compile-time structures
1985: Digitalk (Methods → Smalltalk/V)

512KB IBM PC or clone
1985: Merlin 1

68000
512KB DRAM
black and white video (6845)
serial, then parallel interface to host PC
1986: Swamp

bytecodes
Bitslice
University of Toronto

31 Postive 31 bit integer 0
00....
Object pointer
01....
Stack reference
10....
Negative 31 bit integer
11....
1986: AI32

bytecodes
ASIC: 1.3um CMOS at 16MHz, 12.0 x 11.6 mm
Hitachi
1986: Rekursive

Linn
David Harland

40 bit tagged architecture
Totally object-oriented
32 bit word
Compact representation for 32 bit objects
3 microprogrammable ASIC in 1.5um CMOS,
sea of gates at 10MHz

Logik

Sequencer for high
level languages, 299
pins, 60 bit control
word, 18 fields

Objekt

Object-oriented
memory management, 299
pins, 32 bit control
word, 11 fields

Numerik

32 bit ALU, barrel
shifter, multiplier
and registers, 223
pins, 70 bit control
word, 17 fields
1986: Merlin 2 (Inova)

68000
1MB DRAM (32 bit video bus)
EPROM+PAL video logic
4 colors (redefinable)
Ethernet (first in Brazil)
1987: modified Merlin 2

Jecel & Marcos Alexandre Vieira

Merlin 3:
- 68020
- NuBus & ISA bus

Ethernet replaced with 8 bit ISA bus:
- floppy controller
- hard disk controller
1988: Active Book

ARM2as 8MHz + 1Mb RAM + 2Mb ROM (I think)
Monochrome 600@400 LCD
Like a large tablet PC of current era
Tim Rowledge did port of Eliot Miranda's BrouHaHa plus many extensions.
Underlying OS was port of Helios, a unixish OS derived from TriPos.
Not sold but almost completed before being killed by ATT to remove competition for their (dreadful) GO/Penpoint/EO system.
Full Smalltalk-80 v2 system, rather good performance. Used copy-on-write to allow keeping most objects in ROM so that RAM usage was minimized.
ARM2as was custom variant designed by ABC to be a 'static' part. Stopping the CPU clock would simply halt everything, ready to continue later.
1989: J-Machine

Bill Dally
1989: Mushroom

Mario Wolczko

Ifor Williams
1990: Object Oriented Processor

“scientific initiation” scholarship for Jecel

CAM: Content Addressable Memory

Then the Self numbers were published...

==> 

Low end Sparc
1991: Momenta

386SX 20MHz
4MB RAM
1MB ROM
256KB Flash
40MB HD
DOS+Digitalk Smalltalk
1992: MS8702 MiniSuper

68020
68881 FPU
1MB DRAM
T222 Transputer
ISA bus

16 boards per backplane (4x4 mesh)

4 backplanes per machine (8x8 mesh) = 64 nodes
1993: Merlin 4

ARM 2
VIDC & MEMC
4MB DRAM
Ethernet
PC keyboard

Designed in 1988 when the components were bought and software was written
Do Object-Oriented Languages Need Special Hardware Support?

Urs Hölzle
David Ungar

Figure 4. Instruction mix for SELF and C++ on Richards and DeltaBlue
1997: MediaPad

**StrongARM SA110 206MHz**

Interval Research Corporation, Palo Alto, CA.

Tim Ryan (hardware), Bob Hoover, Bob Alkire, Don Charnley, Paul McCullough, Alan Purdy, Tim Rowledge and Frank Zdybel Jr. with help from Craig Latta, Mike Penk, Kerry Lynn and Phil McBride. Jon Hylands did a web browser and the group included other people

Intended to be a central controller/master remote control for a synchronous network for household media sharing and domestic/industrial control.

No real underlying OS since the Squeak VM was extended to be able to schedule processes for the interrupt system as well as the image.
1997: Squeak On A Chip

Mitsubishi M32R/D
Mits research group
Curtis Wickman
technology demonstration for OOPSLA 1997
Board with 6 chip and PCMCIA memory card
¼ VGA LCD
When? 1998: Tachyon
2001: Internet Appliance

Hyperstone RISC+DSP

Soft Computing Technology
2003: Plurion, dietST, Neo16
2004: SqueakProcessor

```
constant or label

Branch:
next, unext, jumph,
jumpl, dispatch2,
dispatch3, jumpz,
jumpnz

shift
Logic,
add, sub

Offset:
Stack,
Data,
Fetchbyte,
fetchbytemasked,nil, true,
inttag, no memory access

Encoded
Object:
Context,
Method,
Receiver,
pointer

Push,
Store,
Combine,
pop

```

```
Hardware + OS
```

```
inter
prim
obj
```

```
simulated image
image
```

```
.s
```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
2009: SiliconSqueak version 1
2011: SiliconSqueak version 2

SiliconSqueak as an interpreter

bytecode fetch

uCode fetch

address / op read

execute

- method
- ip

- receiver class
- uPC

- SP
- FP
- receiver

- bytecode cache
- uCode cache
- 4KB

- stack cache
- data cache
### SiSq v2 Microcode

<table>
<thead>
<tr>
<th>N:</th>
<th>D or S = 111 means push or pop stack, while 000 to 110 indicate SP-0 to SP-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 next</td>
<td>0001 STR</td>
</tr>
<tr>
<td>0001 return</td>
<td>0010 PSH</td>
</tr>
<tr>
<td>0010 skipOnZero</td>
<td>011 INR</td>
</tr>
<tr>
<td>0011 skipOnOne</td>
<td>100 ADR</td>
</tr>
<tr>
<td>0100 PICuPC</td>
<td>110 SBR</td>
</tr>
<tr>
<td>0101 PIC2</td>
<td>111 ADK</td>
</tr>
<tr>
<td>0110 fetch8</td>
<td>111 SBK</td>
</tr>
<tr>
<td>0111 tagAndFetch8</td>
<td></td>
</tr>
<tr>
<td>1000 fetch1</td>
<td></td>
</tr>
<tr>
<td>1001 fetch2</td>
<td></td>
</tr>
<tr>
<td>1010 fetch3</td>
<td></td>
</tr>
<tr>
<td>1011 fetch4</td>
<td></td>
</tr>
<tr>
<td>1100 jump</td>
<td></td>
</tr>
<tr>
<td>1101 PICx</td>
<td></td>
</tr>
<tr>
<td>1110 call</td>
<td></td>
</tr>
<tr>
<td>1111 tagOrCall</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>K:</th>
<th>U:</th>
<th>A:</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 POP</td>
<td>000 temporary</td>
<td>00xxx - MOV, INC, NOT, NEG, ADD, ?, ?, SUB</td>
</tr>
<tr>
<td>001 STR</td>
<td>001 instance</td>
<td>01xxx - IOR, NOR, XOR, EQV, AND, ANI, NND, NAI</td>
</tr>
<tr>
<td>010 PSH</td>
<td>010 stream</td>
<td>10xxx - SHR, SHL, ASR, ROT, MUL, DIV, UML, UDV</td>
</tr>
<tr>
<td>011 INR</td>
<td>011 special</td>
<td>11xxx - EQL, CRY, MIN, OVF, LTN, LEQ, ULT, ULE</td>
</tr>
<tr>
<td>100 ADR</td>
<td>100 literal</td>
<td></td>
</tr>
<tr>
<td>110 SBR</td>
<td>101 immediate</td>
<td></td>
</tr>
<tr>
<td>111 ADK</td>
<td>11x ALU Matrix</td>
<td></td>
</tr>
</tbody>
</table>

bit 15 = 1 means the 16 bits after the instruction are used to modify uPC
ALU Matrix Coprocessor

- From datapath to datapath
- From ring network
- To ring network
- 64 registers
- 64 sequences of instructions
- 8 bit ALU
- To 4 neighbors
- op rdest ldest src1 src2
2014: Silicon Squeak version 3
2017: Morphle Logic

- **000**: blank
- **001**: horizontal connection
- **010**: vertical connection
- **011**: crossover
- **100**: column match 0
- **101**: column match 1
- **110**: row match 0
- **111**: row match 1

<table>
<thead>
<tr>
<th>AB</th>
<th>AB</th>
<th>AB</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>11</td>
<td>22</td>
</tr>
</tbody>
</table>

- **S**: 0
- **Y**: 1

<table>
<thead>
<tr>
<th>S</th>
<th>S</th>
<th>S</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
2018: RISC-V with SiSq extension

Xsisq
- bytecodes
- stack
- object memory
- PIC
Alan Kay’s recipe for research:

1973
- 3D atomic printing,
- Infosphere

1988
- Grains of computational substrate, IoT

2003
- 3D atomic printing, Infosphere

2020
- Alto

2020
- FPGA

2035
- Wafer Scale
Alto2020: SiliconSqueak version 5

Network Interfaces, I/O, Memory Controllers
NoC (Network on Chip)

- eXternal Flow Processors
- Control Flow Processors
- Data Flow Processors (8 bit ALU+ Registers)
- Dataflow schedulers
- Morphle Logic
SiliconSqueak version 5

Instruction is several LEB128 encoded numbers ending in control flow codes

Equivalent SiSqv3 instruction is 64 bits long
SiliconSqueak version 5

t2 := ((i5 - 1) / (t2 + 100) + (i1 * t3)

load i
5

load lit
0

load t
2

load lit 1

load I
1

load t
3

Frame 0

0 1 2 3 4 5

Frame 1

0 1 2

/ cpy

Frame 2

0 1

store t
2

Frame 3

0 1
Thanks!

Questions?