Keeping old computers alive for deeper understanding of computer architecture

Hisanobu Tomari and Kei Hiraki
The University of Tokyo

Background

- There are a number of options and trade-offs for designing a computer system
  - Instruction set design/instruction encoding
  - Architectural registers
  - Word length, ...
- Different implementation examples help better understanding

Showing Different Implementations?

- Computers students can access are limited.
  - x86_64, ARM
- Nice to have more examples such as:
  - Alpha, SPARC, MIPS, PA-RISC, etc.
- These systems are disposed and not available

Previous Approach: emulators/simulators

- Accuracy is dubious/difficult to evaluate
  - Temporal accuracy: execution time differs from real machine
  - Result accuracy: execution result differs from real machine
- Legal problems (firmware, operating system)
- Difficulty of full-system simulation

- Often easier to repair/restore old machines

Our Approach

- Restore old systems and allow students to access them
  - Help students understand computer architecture better by using wider variety of computer systems
- Enabling students to
  - Get interested in computer architectures
  - Witness how computers have settled to what they are today

Teaching Context

- Target: third-year undergraduate students in an information science department
- Students have knowledge of programming in assembly (PowerPC)
- Students implement original processor on an FPGA in the next semester [Sugawara, 2004]

Course Overview

- 13 lectures in a course
  - Each lecture is 90 minutes long
  - ~8 lectures are used to explain concepts such as:
    - Pipelining, Branch Prediction, Out-of-order Execution, Speculative Execution
  - The other classes are students' presentations

Assignment Details

- Students are divided into 10+ teams
- Each team
  - has 2-3 members
  - is assigned to an instruction set architecture
  - talks about characteristics and features about the ISA after reading the manual
  - writes a program that calculate an inner sum of two vectors, in assembly

Assignment Objective

- Grasp different design goals through learning about different processor implementations
- Learn to read the processor manuals

- Learn to like computer architectures

Keeping Historic Systems Alive

- Most museum-type efforts are focused on storing non-working computers in shelves
  - impossible to verify program for them
- With working systems one can measure
  - power consumption
  - performance using new compiler technique
  - performance with new benchmark
  - other parameters as they become important

Restoration

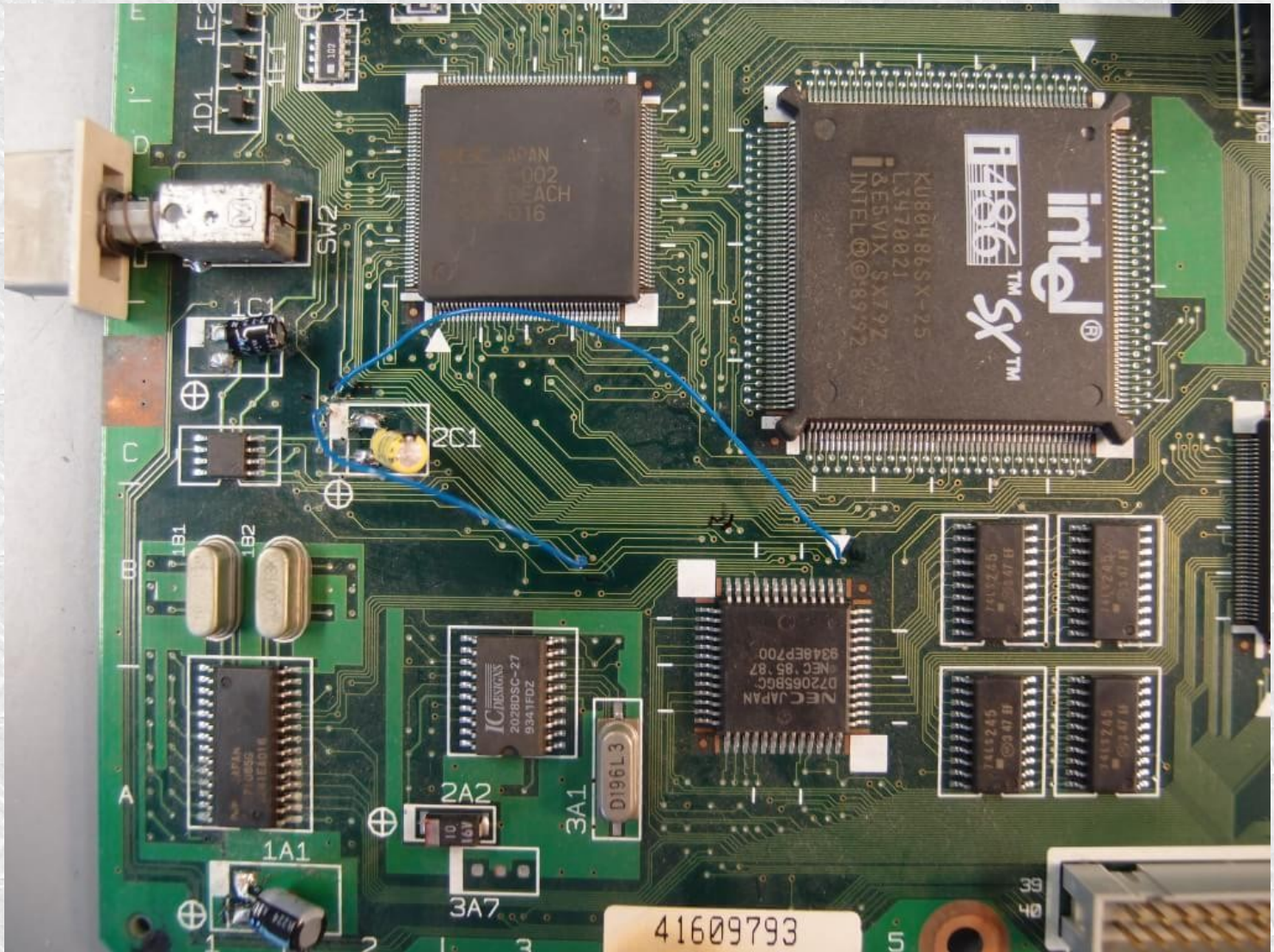
- Removing the plastic mold

Restoration

- Replacing capacitors

Restoration

- Reworking damaged PCB



- So far, >250 systems have been kept in working condition
- Instruction set architectures:
  - Alpha, ARM, i860, IA-64, 68K, MIPS, PA-RISC
  - PowerPC, SH, SPARC, SX, VAX, x86, x86_64
  - Z80, 6502, 6809, …

NEC SX-6i

Commodore Amiga 500

Students' slides

VAX instruction set
(following slides are prepared by Yuichiro Oyabu)

Instruction formats

operation code + operand specifier
(address mode + additional information)

addressing mode

opcode (1 or 2 byte(s))

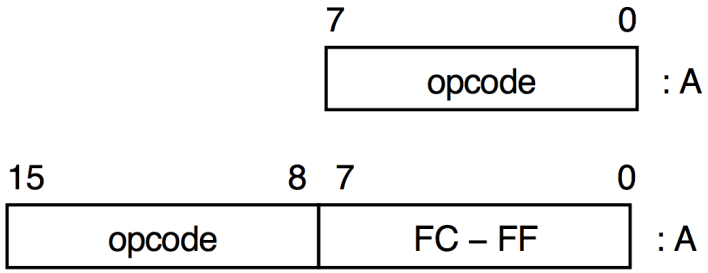


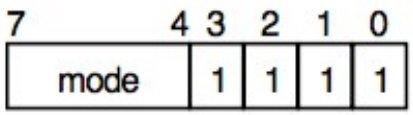
Table 8–6 Program Counter Addressing

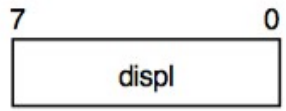
Hex	Dec	Name	Assembler
8	8	Immediate	I^#constant
9	9	Absolute	@#address
A	10	Byte relative	B^address
B	11	Byte relative deferred	@B^address
C	12	Word relative	W^address
D	13	Word relative deferred	@W^address
E	14	Longword relative	L^address
F	15	Longword relative deferred	@L^address

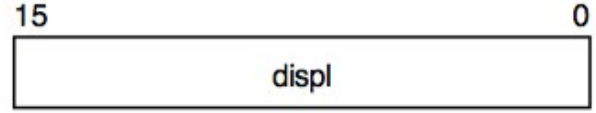
Table 8–5 General Register Addressing

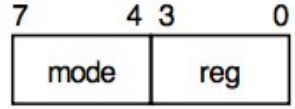
Hex	Dec	Name	Assembler
0–3	0–3	Literal	S^#literal
4	4	Indexed	i[Rx]
5	5	Register	Rn
6	6	Register deferred	(Rn)
7	7	Autodecrement	-(Rn)
8	8	Autoincrement	(Rn)+
9	9	Autoincrement deferred	@(Rn)+
A	10	Byte displacement	B^D(Rn)
B	11	Byte displacement deferred	@B^D(Rn)
C	12	Word displacement	W^D(Rn)
D	13	Word displacement deferred	@W^D(Rn)
E	14	Longword displacement	L^D(Rn)
F	15	Longword displacement deferred	@L^D(Rn)

Branch Mode Addressing









“orthogonality”

- Independence of instruction type and addressing modes
  - All addressing modes are accessible by all inst's

Easiest example：mov inst.
MOVL R1, R2　　register
MOVL(R1), (R2)　　register indirect

On powerpc each instruction is encoded together with addressing modes

Actual code: inner product

```
#NO_APP
        .file   "test.c"
        .text
        .align 1
.globl main
        .type   main,
@function
main:
        .word 0x0
%sp     movab -812(%sp),
        clrl -8(%fp)
        clrl -12(%fp)
        jbr .L2
```

※will replace with hand-written one later

```
.L3:
        movl -12(%fp),%r0
        moval 0[%r0],%r0
        addl2 %fp,%r0
        movl -412(%r0),%r1
        movl -12(%fp),%r0
        moval 0[%r0],%r0
        addl2 %fp,%r0
        movl -812(%r0),%r0
        mull2 %r1,%r0
        addl2 %r0,-8(%fp)
        incl -12(%fp)
.L2:
        cmpl -12(%fp),$99
        jleq .L3
        clrl %r0
        ret
        .size   main, .-main
```

Data types and reαs

・ データタイプ : 14種類

Byte, Word, Longword, Quadword,

Octaword, F_floating, D_floating, G_floating,

H_floating, Variable-Length Bit Field,

Character String, Trailing Numeric String,

Leading Separate Numeric String,

Packed Dicimal String

・ レジスタ : 16個の汎用32ビットレジスタ

R15:PCプログラム・カウンタ

R14:SPスタックポインタ

R13:FPフレームポインタ

R12:APアーギュメントポインタ

R11−R0:汎用

Table 3-1 Special Register Usage

Register	Hardware Use	Conventional Software Use
R0	Results of POLY,CRC; length counter in character & decimal instructions	Results of functions, status of services (not saved or restored on procedure call)
R1	Result of POLYD; address counter in character & decimal instructions	Result of functions (not saved or restored on procedure call)
R2, R4	Length counter in character & decimal instructions	any
R3, R5	Address counter in character & decimal instructions	any
R6-R11	None	any
AP (R12)	Argument pointer saved & loaded by CALL, restored by RET	Argument pointer (base address of argument list)
FP (R13)	Frame pointer saved & loaded by CALL, used & restored by RET	Frame pointer; condition signalling
SP (R14)	Stack pointer	Stack pointer
PC (R15)	Program counter	Program counter

Characteristic instruction

- POLY—polynomial

e.g. calculation of sin(1)（x/1!-x**3/3!+x**5/5!で近似）

```
POLYF #1, #5, PTABLE
PTABLE:
        .FLOAT 0.008333
        .FLOAT 0.0
        .FLOAT 0.0
        .FLOAT -0.166666
        .FLOAT 0.0
        .FLOAT 1.0
        .FLOAT 0.0
```

- Queues and stacks—makes it easier to write epilogie and prologue
- CRC—error detection in one instruction
- EDITPC—function for editing

VAXstation 4000 60

NetBSD has been set up to the student's preferences

Student's slides: Remarks

- Students presents about:
  - Instruction set and encoding
  - Data types
  - Registers
  - Characteristic instructions/features
- Writes program to calculate inner product
- Verifies their interpretation of manuals using real system
- Evaluate the performance of real implementation

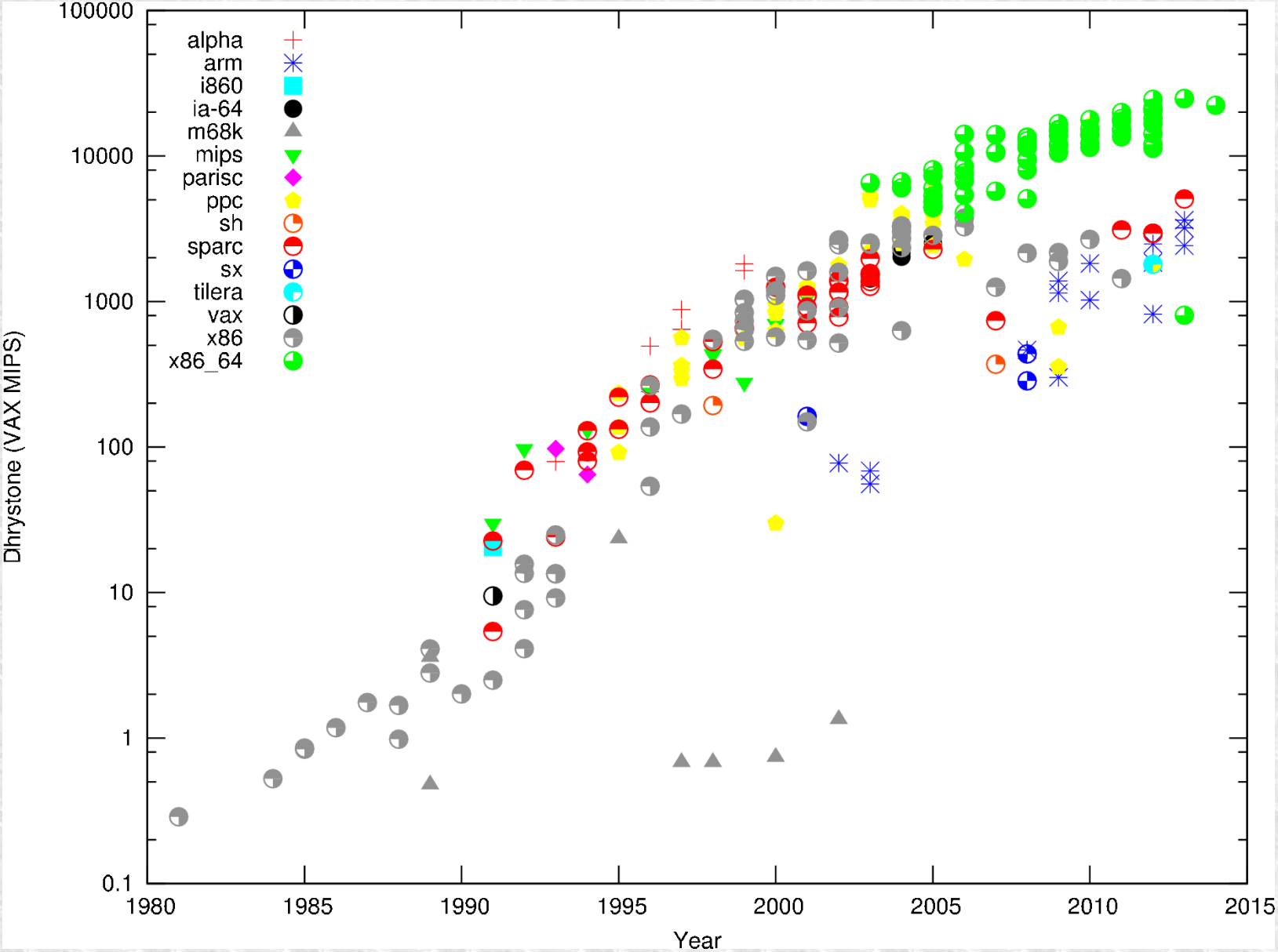
Conclusions

- Old computers help students understand concepts of computer architectures better
  - Draws more attention than emulators
- Keeping them in working condition takes a lot of effort
- Students interactively feel the difference of processing speed in person

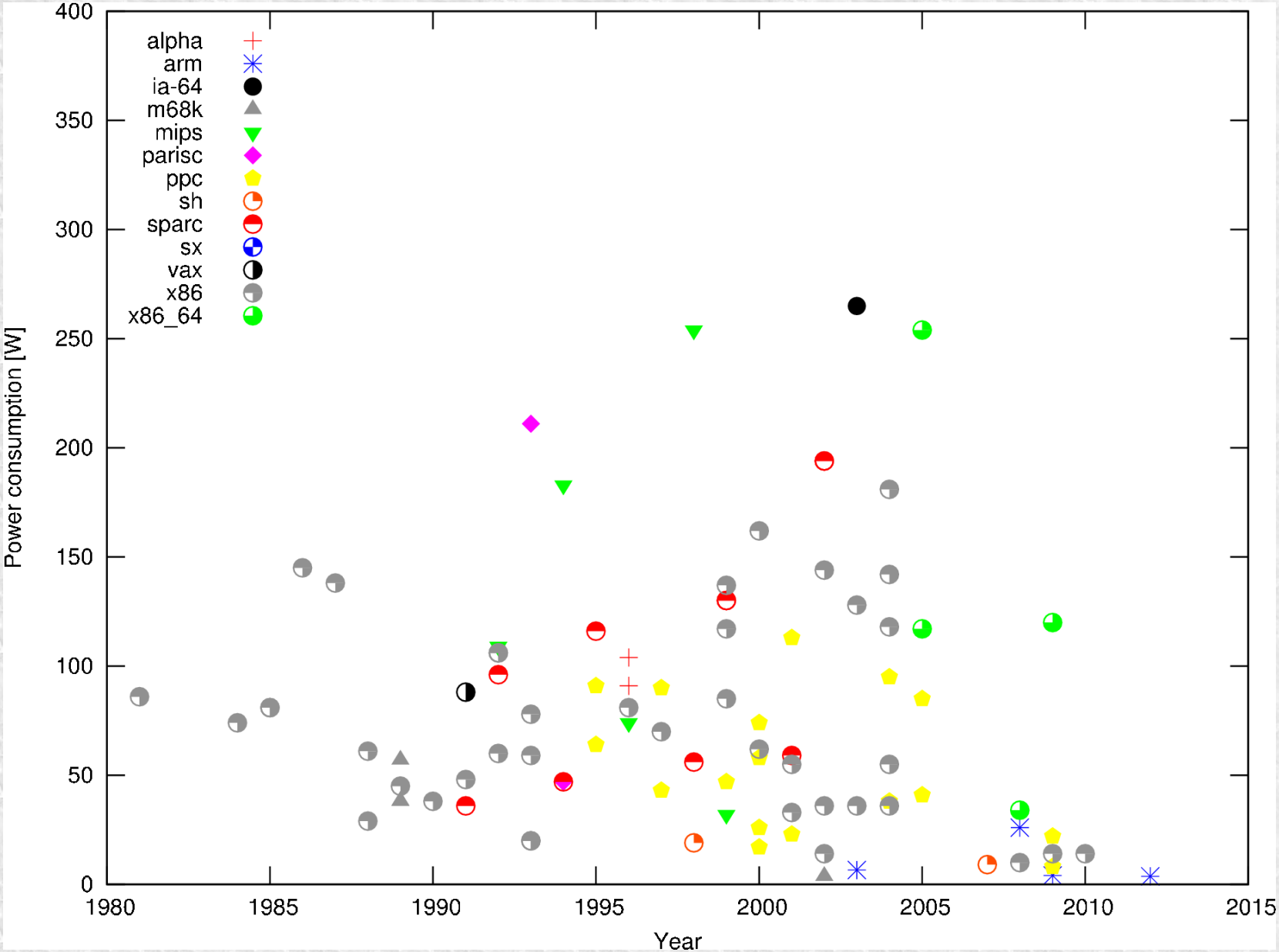
Measurement results

- As we repair old systems, benchmarks are done
- Aids quantitative understanding of computers through
  - Performance
  - Power consumption
  - Applying different evaluation methods
- Same/similar compiler, optimization flags can be used across wide generation of systems

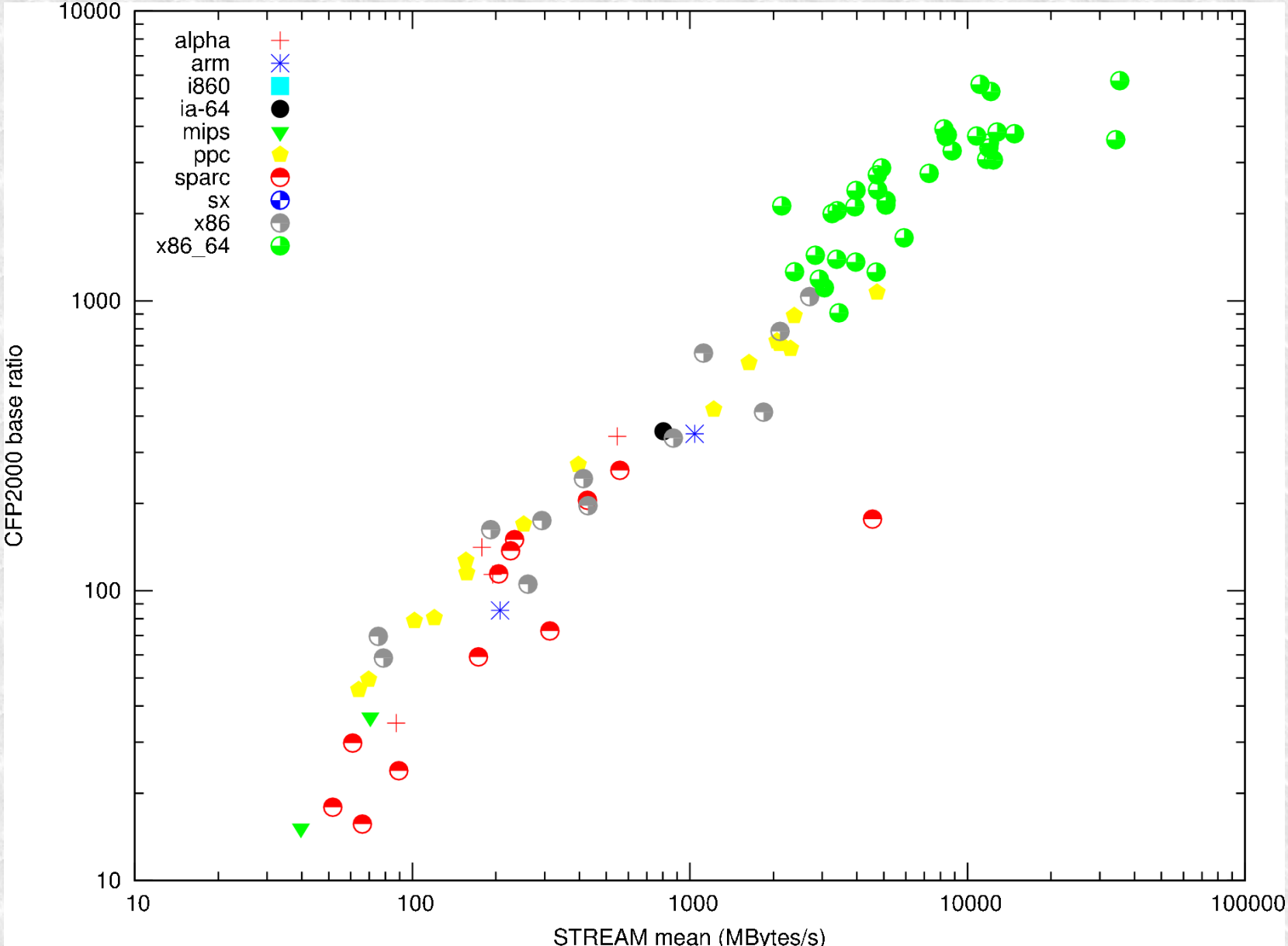
Dhrystone, VAX MIPS



Power consumption of Single-socket systems



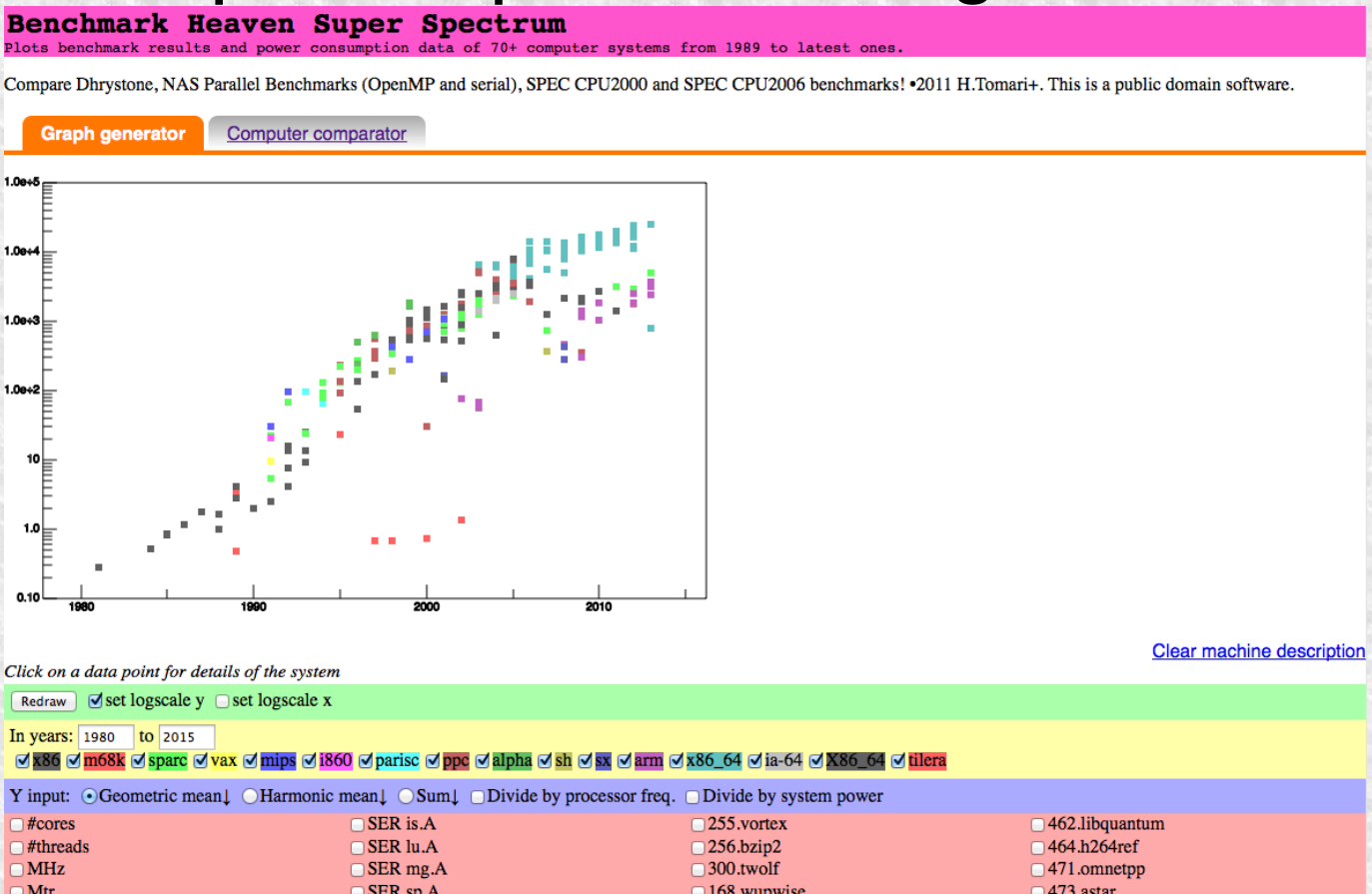
STREAM - CFP2000



There are much more data

- Web interface to compare results has been developed and deployed at

http://computer-zoo.org/



Wishlist:

PDP-11
64b PA-RISC
Transputer (evaluation kit preferred)