SIMH: Forward... Into The Past

Bob Supnik, VP, Sun Microsystems

Contents

- An introduction to SIMH
- Rationale
- SIMH's development history
- The role of the Internet
- SIMH design principles
- Building a simulator
- The computer history ecosystem
- Demonstrations
- Going forward

What is SIMH?

- SIMH is an Internet-based collaborative project focused on preserving computers (and software) of historic interest via simulation
- SIMH consists of
  - A portable application framework for implementing simulators
  - Portable implementations of 20+ simulators on this framework
  - Demonstration software to run on these simulators
  - Papers and presentations documenting interesting facts and tidbits gleaned from the simulators
- On the Web at http://simh.trailing-edge.com

Portability

- SIMH runs on
  - X86 Linux, NetBSD, OpenBSD, FreeBSD (gcc)
  - X86 Windows 95, Windows 98, Windows 2000, Windows XP (Visual C++ or MingW)
  - WindowsCE
  - Mac OS/9 (Codewarrior) or OS/X (Apple Development Tools)
  - Sun Solaris (gcc)
  - HP/UX (gcc)
  - AIX (gcc)
  - Alpha Unix (DEC C)
  - VAX/VMS, Alpha/VMS, IA64/VMS (DEC C)
  - OS/2 (gmx)

Scope

- SIMH implements simulators for
  - Data General Nova, Eclipse
  - Digital Equipment Corporation (DEC) PDP-1, PDP-4, PDP-7, PDP-8, PDP-9, PDP-10, PDP-11, PDP-15, VAX
  - GRI-909
  - IBM 1401, 1620, 1130, System/3
  - Hewlett-Packard (HP) 2116, 2100, 21MX
  - Interdata (Perkin-Elmer) 16b, 32b architectures
  - Honeywell H316/H516
  - MITS Altair 8800, both 8080 and Z80 versions
  - Royal-McBee LGP-30, LGP-21
  - Scientific Data Systems SDS-940
- More than two dozen machines in all
- Rather biased towards East Coast USA...

Software

- With SIMH, you can run
  - PDP-1 Lisp and DDT, early interactive systems
  - PDP-11 Unix V5, V6, V7, the earliest extent releases of Unix (V1 to V4 are lost)
  - Interdata 7/32 Unix V6, the first port of Unix (and the first port to a 32b system)
  - PDP-10 TOPS-10, TOPS-20, ITS
  - PDP-11 DOS, RT-11, RSX-11M, RSX-11M+, RSTS/E
  - PDP-15 ADSS-15, F/B-15, DOS-15, DOS/XVM
  - PDP-8 OS/8, TSS-8, ETOS, DMS
  - VAX/VMS, VAX/Ultrix, VAX/BSD, VAX/NetBSD
  - Nova RDOS, Eclipse AOS
  - HP DOS, RTE-III, RTE-IV
  - MITS Altair CP/M, DOS
  - System/3 SCP, CMS

Rationale

- Computing’s past is disappearing
  - As machines are scrapped, software and documentation is thrown out, media becomes unreadable, and industry pioneers die
- The Santayana principle
  - “Those who do not study the past are condemned to repeat it”
  - Hardware and software engineers re-invent the breakthroughs (and mistakes) of the past because they don’t know what they are
  - What’s the difference between optimizing for the 8KW of a PDP8 and the 8KW of a microprocessor’s first level cache? (Answer: no one does the second)
- The relative lifespans of hardware, software, and data
  - Hardware and architectures come and go
  - Software and data have much longer duration: 15-50 years
  - What will run the BART signs when the last PDP-8 breaks down?

No, No, Why *Those* Systems?

- The LGP-30 was the first computer I ever saw
- The IBM 1620 was the first computer I ever programmed
- The PDP-7 and the PDP-8 were the first computers I wrote complete projects for
- The Nova was the first computer I did a complete system design for
- The GRI-909 was the weirdest architecture I ever wrote code for (***one*** instruction)
- I've always had a sneaking fondness for 24b machines (there were never any successful ones)
- I worked at DEC for 22 years

And Why Did You *Really* Write SIMH?

- Larry made me do it
  - Larry Stewart, DEC Research, pointed out in 1993 that computing's past was being lost and suggested I do something about it
- I needed to graduate from programming in assembly code and microcode
  - With Alpha, availability of VAXen was beginning to decrease
- It seemed like a good idea at the time
  - I hadn't done a major software project since porting Dungeon to the PDP-11 (in the late 70's)
  - I needed an excuse to learn C
- Besides, how difficult could it be to write a simulator?
  - 11 years and 125K lines of code later…

Project Goals

- Goal: make computers and software of historic interest accessible to a broad technical population
  - Simulators, rather than restored hardware
  - Highly portable (at least VMS, UNIX, and Win32)
- Starting point: MIMIC, an RTL simulation system from the late 60's and early 70's
  - Theft (of one's own prior work) is the highest form of productivity
- Initial targets: well documented minicomputers
  - DEC PDP-8, PDP-11
  - Data General Nova
- One man, one code base
- Then came the Internet...

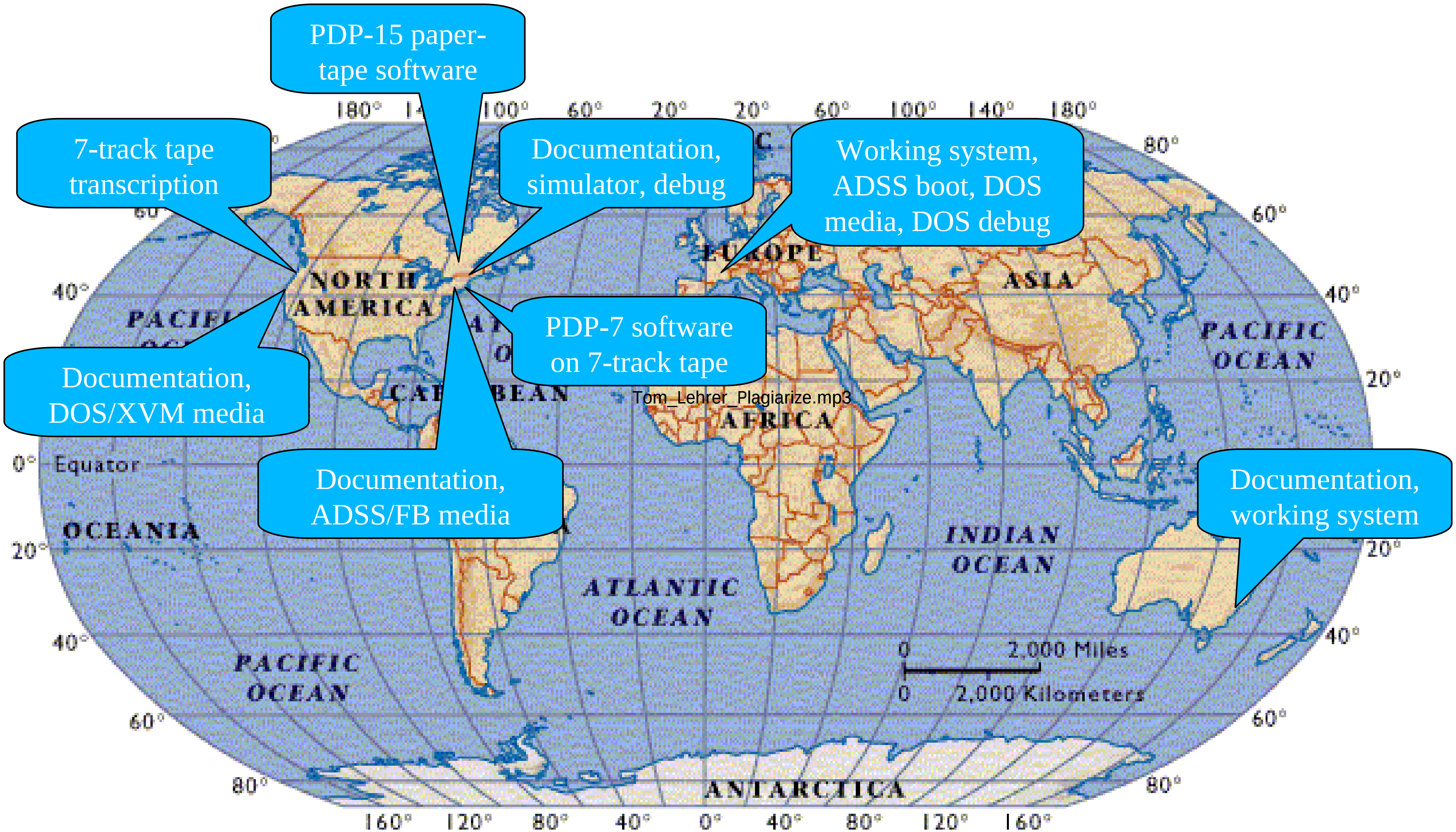
SIMH And The Internet

- The Internet has turned out not be a “global city” but a million global villages, and computer history is one of its communities
- Initial contacts through newsgroups and mailing lists
- The Web made the activities of collectors and hobbyists visible and accessible
  – Document repositories
  – Simulation systems
  – Software stashes
- As a result, SIMH morphed from a one-person hobby project to a collaborative effort of more than 30 people
- Most of us have never met IRL

The Computer History Ecosystem

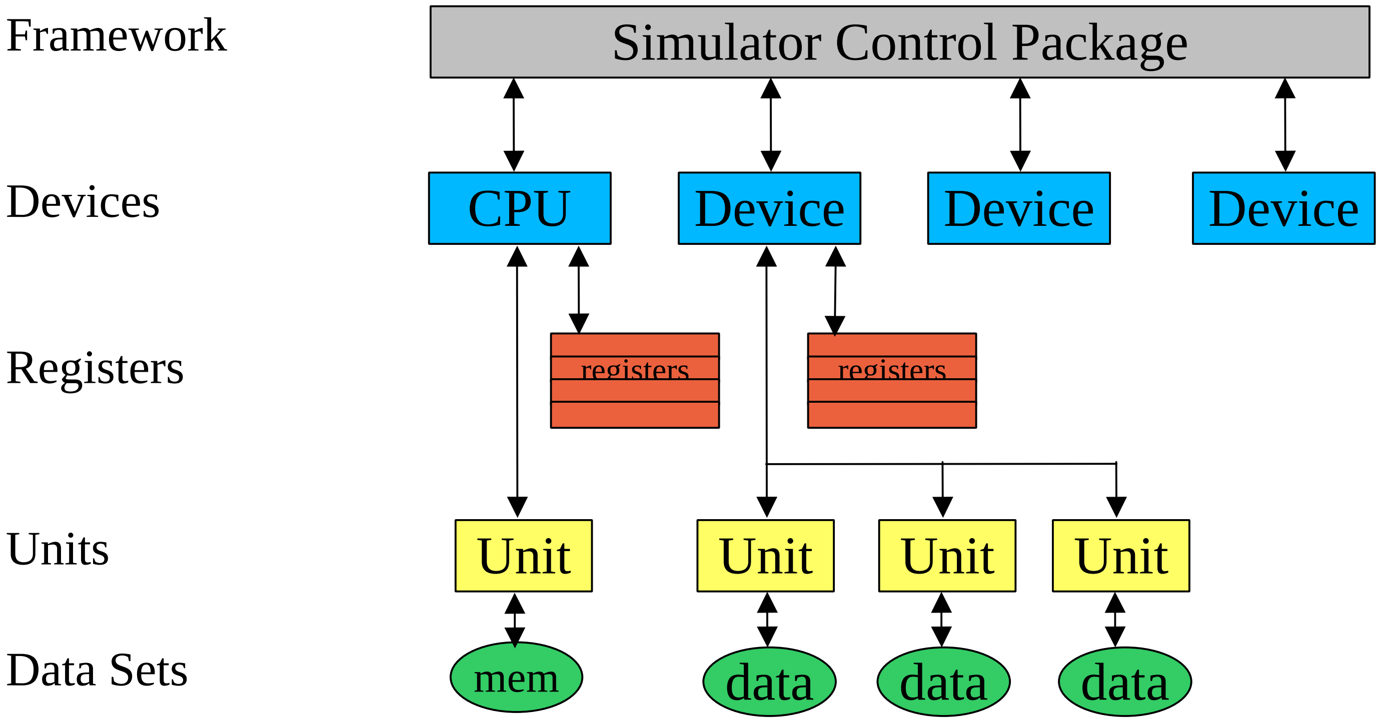
- Private collectors
  - Sometimes there’s no substitute for real hardware
- Document archivists
  - Bringing the written word on-line
- Simulator writers
  - SIMH, MAME, Hercules, CyberCray, and many, many others
- Restoration projects
  - Rhode Island Computer Museum PDP-9, La Cite des Sciences et L'Industrie PDP-9, Computer History Museum PDP-1 and 1620
- Institutions
  - Computer History Museum, RICM/RICS
- Ebay
  - Ultimately, everything is put up for sale

SIMH and The Internet: Recovering Software for the 18b PDP's



SIMH Design Principals

- Simulators are collections of *devices* (the CPU is just a device that executes instructions)
- Devices contained named *registers*, which hold state, and numbered *units*, which contain data sets



Design Principals, continued

- Data sets are mapped into a uniform set of host system containers
  - Containers can be in memory (arrays) or on disk (files)
  - Containers are constrained to “natural” size boundaries (e.g., a 12b memory is mapped as a 16b array or file)
- Asynchronous behaviour is modelled explicitly
  - Time tracked in convenient units (nanoseconds, instructions, etc)
  - Device events are scheduled for “future time”
  - Simulator calls a device event handler at appropriate point
- Common devices classes are implemented through libraries that hide host OS dependencies
  - Libraries for disks, tapes, terminal multiplexers, Ethernet
  - Future extensions will support graphics, “raw” device access

Writing A Simulator: A Three Step Program

- Step 1 – research
- Gather as much documentation as possible
  - Primary documentation (maintenance manuals, print sets, microcode listings) are preferable to secondary sources (handbooks, user's guides, prior simulators)
- Make contact with actual users
  - Folklore can be as important as the printed word
- Gather and transcribe required software
  - Diagnostics, operating system(s), application code
- RTFM (both the target system's and SIMH's)
- The Internet provides a wide variety of starting points for gathering information

Writing A Simulator, continued

- Step 2 – implementation
- Critical design decisions
  - How will instructions be decoded and executed?
    - Modern computers are fast, don't waste time on optimization
  - How will the I/O subsystem be modelled?
    - Typically, the more accurately the better
  - How will interrupts and exceptions be handled?
    - Typical hardware mechanisms, like microtraps, are easily implemented with **longjmp** and poorly implemented with try-except-finally
  - What debugging facilities should be included?
    - These will be used to debug the simulator, not new programs
- There are plenty of examples to emulate (or borrow)

Writing A Simulator, continued

- Step 3 – debug
  - Hand test cases, to get out the stupid bugs
  - Diagnostics, to get out the straightforward bugs
  - Operating systems, to get the details right
- Successful operating system and application operation is the only real proof of completion
- Operating systems make excellent go/no-go diagnostics, but their reporting mechanisms (crash, hang) leave something to be desired
- A typical “large” simulator seems to have ~100 bugs
  - The last 20 have to be found with operating system software
- Hence, the need for strong simulator debug tools (step, save and repeat, breakpoints, traces, etc)

The Devil Is In The Details

- Writing and debugging a simulator can resemble detective work more than software engineering
  - Hardware documentation may be incomplete
  - Hardware documentation may be misleading or false
  - Software may be incomplete
  - Software paths may be untested
  - Simulated configurations may be untested
  - Software may have undocumented timing dependencies
  - Software may have undocumented hardware dependencies

Making The Right Tradeoffs

- Accuracy is more important than performance
  - Implement a specific system, not an architecture
  - Model the hardware accurately at the “black box” level
  - Follow the microcode (if applicable and available) or the logic prints
  - Include the fine-grain details
- Allow for “real-world” interactions
  - Wall-clocks vs simulated clocks
  - Timing loops and real-time devices
- Be prepared to “scale out”
  - Users may not be satisfied with real-world speeds and feeds

Demonstrations

- PDP-11 Unix V5 – the earliest extent Unix
- Interdata Unix V6 – the first port, the first 32b Unix
- PDP-10 TOPS-10 – a timesharing bureau on your laptop
- VAX/VMS – 6 9's availability for your PC

SIMH In The Real World

- SIMH has grown steadily in scope, scale, and complexity
  - 1996: 6 simulators; today: 24 simulators
  - 1996: 12b and 16b systems; today: up to 32b (with 64b to follow)
  - 1996: 2 host platforms; today: 18 host platforms
- SIMH is being used beyond the hobbyist community
  - As the development platform for PDP-11 OS development
  - As a replacement VAX platform in a government software development program (reduced build cycles from 135 minutes to 14 minutes)
- Future releases will allow more “real world” interactivity
  - Graphics
  - Access to additional real devices on the host system

What You Can Do

- Check your attic (or your father’s… or grandfather’s)
  - Lots of equipment, media, software, documentation still in private hands
  - The greatest risk is simple discarding of “unimportant” artefacts
  - If in doubt, consult the Computer History Museum
- Write (or adopt) a simulator
  - Lots of interesting machines still to do
  - Lots of simulators sitting in unfinished or untested state
  - You never forget your first computer (much as you'd like to)
- Get *involved*!
  - Preservation of computing’s history depends more on individuals than on institutions or governments