UTX/32 and the art of fire breathing

a long forgotten chapter in UNIX[1] history

Geert Rolf
Former sysadmin (1987-1991), owner of a Gould PowerNode 6040 (1994-) and
contributor to the SIMH/sel32 project (2019-2023).
Winssen, The Netherlands

Abstract

UTX/32 is an operating system that dates back to the 1980s and was designed for Gould/SEL 32-bit machines. Competing with DEC VAX 11/780, the Gould PowerNode 6000 and 9000 systems were very powerful and cost effective. Marketing labelled them fire-breathing dragons.
This paper takes you back to the 1980s. Back to the wars fought in the Unix world and the position UTX/32 held. Are you a BSD or a System V believer? UTX offers the choice of whatever your favourite is.
Last year a simulator for the SEL-32 systems, SIMH/sel32, was released. The legacy of a long-gone piece of UNIX history has revived, running UTX/32 and applications on today's computers. After about three decades, the fire breathing dragons have returned.

Introduction

Systems Engineering Laboratories (SEL) was founded in 1961 and based in Fort Lauderdale, Florida. SEL designed and built minicomputers focussed on real-time applications. A range of computers, the SEL 800 series, was introduced.

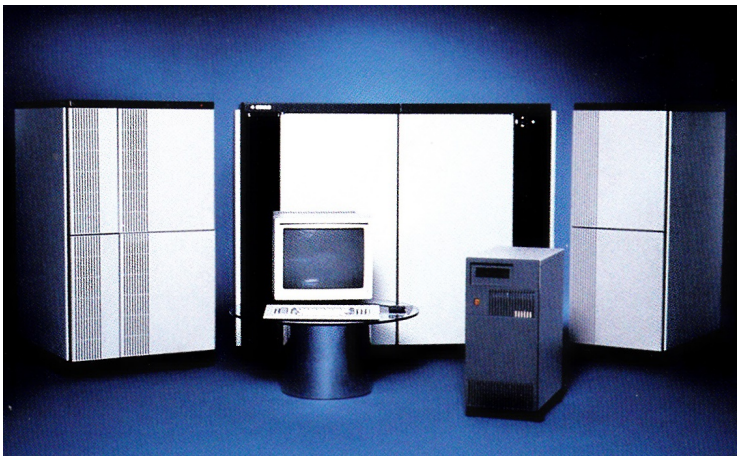
The early models were 16 bit and later models were 24 bit systems. In 1970 SEL entered the 32 bit market by introducing the SYSTEMS 85 and 86 both wire-wrapped systems with core memories. These were batch oriented systems aimed at real-time and data acquisition applications. By 1975 SEL launched a multitasking system: the SEL-32/55. At time of introduction, the SEL-32/55 had one competitor: the Interdata[2] 7/32[3], which was launched in 1974. Digital Equipment Corporation (DEC) introduced its first 32 bit machine, the VAX 11/780, in October 1977. Development continued with the SEL-32/75 with core and the 32/77 with MOS memory. MPX/32, the Mapped Programming Executive, was developed for the 32/75. The first dual CPU system was introduced with the model 32/7780.

In 1981 SEL was taken over by Gould Electronics and renamed Gould Computer Systems Division (Gould CSD). The SEL-32 series was rebranded Concept-32. In 1988 Gould Electronics was acquired by Nippon Mining a Japanese company, but, due to a significant installed base at the U.S. Department of Defense, was forced to sell off

1 UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd
2 Interdata was taken over by Perkin-Elmer in 1973 and spun off as Concurrent Computer Corporation in 1985.
3 The University of Woolongong ported UNIX v6 to the 7/32. AT&T Bell Labs ported to its successor, the 8/32.

Gould CSD, that became part of Encore Computer Company, a U.S. company. Encore, based in Marlborough (MA), produced the MultiMAX multiprocessor systems based on National Semiconductor 32032 processors. On the MultiMAX they offered a choice between UMAX 4.2 and UMAX V[4].

In 1983, long before the merge with Encore, two Concept-32 models formed the base for the development of the PowerNode series 6000 and 9000[5]. The Concepts could run a port of UNIX V7, but lacked virtual memory. The PowerNodes got the hardware to support paging and virtual memory..



Left to right in background: PN 6080 dual processor, PN 9080 dual processor, PN 6040 single processor. Foreground: Gould Engineering Workstation, in fact a SUN-3.

The PowerNode 6000 was based on the Concept 32/67, a TTL[6] based system. The PowerNode 9000 originates from the Concept 32/97 and was ECL[7] based. Years later there was a Single Slot 6 "SS6", which was a single board implemention of a Concept 32/67 and a PowerNode 6000 selectable by a jumper setting.

The main competitor: DEC VAX

|d|i|g|i|t|a|l| VAX 11/780 at TNO (NL)
(apparently running VAX/VMS)

In the early 80s UNIX rolled out in the academic world. It was very popular on Digital Equipment Corporation (DEC) VAX 11/780 and the smaller and cheaper VAX 11/750.

In the world of Universities, UNIX typically entered the computer room on a DEC VAX. This was largely due to the work done by the Computer Systems Research Group (CSRG) at the University of Berkeley. Their port of UNIX to the VAX featured demand paging/virtual memory, a fast filesystem and networking for IP and XNS, including the socket layer.

The Concept-32 series[8] originate from 1975. These computers were successfully targeted for the real-time market and did not run UNIX in the beginning. The Concepts running the MPX/32 real-

4 UMAX 4.2 and UMAX V: a choice between BSD and AT&T flavours of UNIX.
5 Internally called "V6" and "V9" for their capability to handle virtual, demand paged memory.
6 Transistor-Transistor Logic: basic building blocks for digital electronics. CMOS technology adapted too.
7 Emitter-Coupled Logic: fast, low gate delays, and high fanout capacity. Speed above power consumption.
8 Sold as SYSTEMS-32 in early years and later renamed Concept-32, this is the SEL-32 family of systems.

time operating system, were well known in applications such as simulators for aircraft and power plants, real-time process monitoring, control and data acquisition in science and industry.

More about the hardware. [9]

The CPU of a Concept 32/67 or a PowerNode 6000 is implemented on three boards[10]. The Floating Point Accelerator takes another two. The boards were quite huge: 15” x 18” (38 x 46 cm).

The SEL-32 is designed around a high speed bus, the SelBUS operating at 26.6 MB/sec. Designers had an IBM background and introduced a channel concept in this range of minicomputers. Other IBM influences: big endian byte order and System/360 style floating point.

The controllers on the SelBUS are “channels”: they process a sequence of commands that are similar for disk, tape etc. Surprisingly, the bootstrap code is exactly the same for a tape or a disk, ignoring the differences between tape drives and disks.

An important channel is the I/O Processor (IOP). This channel serves the console panel and terminal, the clock, and interfaces on the Multi Purpose (MP) bus, like a floppy disk controller.

Channels process a Command Chain with a list of I/O operations. Part of the list can be a Data Chain: telling where to get or deliver data related to one single I/O operation.

A striking example of a Data Chain is in the handling of network traffic. In a BSD-style kernel, protocol messages are stored in a linked list of *mbufs* that help avoid copying data when packing or unpacking a protocol message[11].

The ethernet controller can read a packet from the network and store the data in a set of linked mbufs. For each mbuf used, it needs one I/O Command Double word in the Data Chain, each one specifying how many bytes are to be stored in the target mbuf.
In this example the Data Chain allows the data from one single DMA[12] I/O command to be delivered at scattered memory locations.

9 Typically, Unix guys ‘n girls don't know much about the hardware they use. With few exceptions: sizeof(int) and for deeper level the byte-order of the machine word.
10 The CPU of a Concept 32/97 takes nine boards to implement due to lower density of ECL components.
11 Adjusting the offset and the data length in an mbuf is used to strip off a header. Messages, that are to be sent, are placed at the end of an mbuf, so any preceeding header can be added at the front without copying the data. When exceeding an mbuf one can add a new one in the chain. In UTX/32 mbufs can store max 232 bytes.
12 DMA, Direct Memory Access: controller puts/takes data into/from memory without CPU intervention.

CONCEPT 32/67

Gould Concept 32/67 part of DC9 fixed base flight simulator, SVS Flyfaglinja, Lillestrøm (NO), 2008

POWERNODE 6040

Gould PowerNode 6040, Library for the Blind, Nijmegen (NL), 1989

The optional second CPU.

The second CPU, first appeared in the SEL 32/7780, is called Internal Processing Unit, IPU. The IPU and the CPU differ in a few details:

- The trap and interrupt vectors are at different locations.
- In a standard configuration CPU and IPU have access to all of the memory.
- Both CPU and IPU have cache memory. A range of addresses in the main memory can be excluded from the cache to ensure integrity between processors.
- Two instructions set or clear a bit in memory and include a memory lock to implement the set and release of a semaphore.
- All I/O instructions are trapped at the IPU. Operating system can handle the I/O at the CPU and when done execution may continue at the IPU.
- A specific Signal IPU (SIPU) trap is used for signalling events from CPU to IPU and vice versa.

Where were Concepts and PowerNodes used in The Netherlands?

- NIKHEF, research institute for nuclear and high energy physics. (32/87, 9080, NP1)
- Bureaus voor Rechtshulp (12x 6040)
- Hogeschool Den Haag (9080)
- Libraries for the Blind (2x 6040)
- KLM: Schiphol-Oost: about 7 simulators using Concepts most model 32/87.
- KLS/KLM Eelde: Airbus A-310 by CAE: Concept 32/8780.
- Philips Semiconductor Nijmegen: Concept
- Fokker Aircraft: Concepts embedded in flightsimulators for F-70 and F-100.
- Fokker Space: 32/67 for ESA project.
- Fokker Control Loading: Concept 32/27s
- TU Delft, air- and space technology.
- PGEM, power network control by Siemens.
- ESA/Estec Noordwijk: Concept 32/87.
- ECN Petten, power plant simulation.
- Friendship Simulation, Beek (4x 32/67)
- NLR: Concept 32/67.
- TNO: Concept
- ...

The dual CPU config is used as master/slave, leaving the main CPU in charge of I/O. A front panel switch[13] allows the selection of which of the two processors is to be used as CPU or IPU.

13 Jumper selectable on the 32/7780. Panel switch on later models like 32/67, 6080 etc.

Concepts 32 can be delivered in custom configurations, even with a second IOP and extra console tty for the IPU. PowerNodes were offered in standard configurations: all memory shared and the IPU without a second IOP or console terminal.

About the software

The port of UNIX V7 for the Gould Concept-32 did not support paging and virtual memory, due to lack of hardware support. The PowerNodes 6000 and 9000 filled that gap.

By 1984, Gould CSD took over Compion, a spinoff company of the University of Illinois' Center for Advanced Computation, involved in development of the operating system and computer language for the Illiac IV supercomputer. This new company in Urbana-Champaign (IL) became Gould's UNIX R&D centre.

In parallel with the hardware development of the PowerNode 6000 and 9000, Gould UNIX R&D developed UTX/32 as a port of BSD 4.2 enriched with a lot of goodies from AT&T and SUN Microsystems. A dedicated focus was on providing tools for a real-time employment for UTX/32. By 1987 UTX/32 v2.1 was released, based on 4.3 BSD.

Features by SUN Microsystems:

A major spin off from the BSD developments has been the founding of SUN Microsystems Inc (SMI). History learns that mentioning "spin off" is ignoring the heavy dispute between the Regents of UCB and SMI about Bill Joy taking 4.2 BSD sources with him. Anyway, SMI contributed a lot to the development of Unix and traces of their work are still in many modern Unix and Linux versions:

- Network Disk (ND) to provide raw disk access over the network for root and swap partitions used by SUN-2 and SUN-3 diskless stations running SunOS 3.x. These workstations could boot from a PowerNode server.
- Network File System (NFS).
- Yellow Pages (YP), also called Network Information System (NIS) to share a.o. password, group, host files in a cluster of centrally managed (work)stations.
- External Data Representation (XDR), a software layer to exchange data over a network between machines with different architecture (a.o. byte order in a machine word).
- Remote Procedure Calls (RPC), a layer of software to support calling functions remotely over the network.

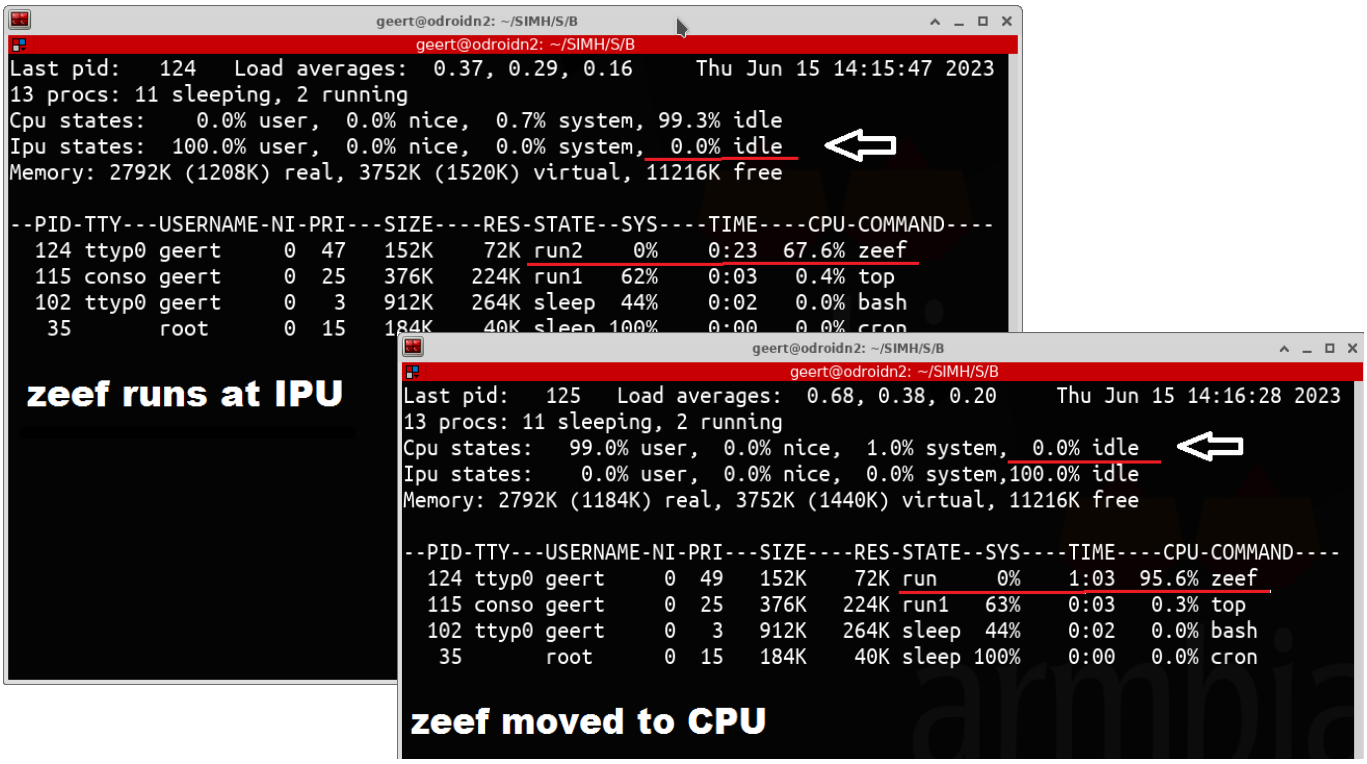
Features in AT&T System V Release 2 that got into UTX/32:

- commands, libraries, system-calls in System V style are supported in UTX/32. Processes in SV environment are marked as such and system calls are handled conform SV standards whenever they differ from BSD calls.
- Selection of PATH is the key of chosing between environments: /usr/5bin and specific place for System V compatible libraries (/usr/5lib) are significant.
- Use of cpio where BSD die-hards prefer tar. Assume it works.
- Basic System V release 2 style IPC features: Messages, Semaphores and Shared Memory.

Support for real-time applications was a big niche for Gould. This was a strong motivation to make the many real-time applications possible at the PowerNodes by providing the necessary real-time features that standard UNIX lacked. An integration of software development and real-time operating environment was the goal one had in mind.

Features of Real-Time-Enhanced UNIX:

- Determinism and control in process scheduling, bypassing the standard scheduling.
- Cyclic scheduling, also bypassing standard scheduling. Cycles divided in Frames.
- High speed context switching: suspend and resume.
- High-resolution event scheduling and execution-time measurement.
- Control over paging and swapping. Lock in memory after all is paged in.
- Deterministic interprocess communication.
- Contiguous files, deterministic transfer times, asynchronous I/O capability.
- Support for the High Speed Data channel for analog and digital input and output.
- Direct I/O including asynchronous I/O operations.
- Access to special purpose memory.
- Connected interrupts. Either directly or indirectly. A driver in userland.
- Access to privileged instructions.
- Process targetting at CPU or IPU. As illustrated below, a benchmark program “zeef”, explained later, is cpu bound and runs on the IPU by default. Per process a target cpu mask is maintained that can be set to determine where a process will run. The facility used here: **settargetcpumask(pid, setmask)** to restrict the process to run on the CPU only.



Gould CSD developed the real-time extensions for their customers in the real-time market.

With a 70% share, the Concept-32s were very successful in the market for Flight Simulator Training Devices. Among others the Concepts were used by CAE, originally known as Canadian Aviation Electronics, the major manufacturer in the market for flight simulators.

Applying a Unix machine in a real-time environment is at the cost of its performance as general purpose time-sharing system. Let your UTX machine control a Full Flight Simulator and at the same time support 40 students who face next week's deadline of their practical exams isn't the right mix. It's only about offering the same environment for development and runtime of a real-time application. Real-time systems are tailored systems…

It is not known how successful this effort has been. The market share in flight simulators was quite impressive, but the number of manufacturers is limited and those were the ones deciding about their development platform.

CONCEPT 32/67

One of two 18 slot SelBUS boxes. From top to bottom:

- FPA two boards
- CPU three boards
- Disk processor, two boards
- ADI analog/digital single board

The other SelBUS box (not on this picture) holds:

- FPA two boards
- CPU three boards
- IOP for clock, console tty and MP BUS
- Tape processor
- 2x IMM memory boards of 1MB each.

An 8 slot MP bus holding Line Printer/Floppy disk interface and one 8LAS for eight serial lines.

POWERNODE 6040

Box with -from right to left- 10 slot SelBUS:

- BTP buffered (mag)tape processor
- HSDP high speed disk processor
- EC Ethernet processor
- ISM 4MB Integrated SelBUS memory
- CPU three boards
- IOP for clock, console tty and MP BUS
- ISM 4MB Integrated SelBUS memory

On the left side: 4 slot MP bus:

- 2x 8LAS 8 line serial cards in MP BUS
- 2x dummy board for airflow

Except for memory size, this 6040 is a relatively small machine compared with the 32/67 on the left.

Benchmarking

By mid 80s many Unix conferences featured a vendor exhibition with a lot of Unix machines. To value and compare the performance of these variety of systems, Andrew Tanenbaum (VU Computer Science) and Teus Hagen (Mathematical Centre) designed two little tests. To connect with 1983 measurements, let's take a closer look at "*Two programs, many UNIX systems*" by Tanenbaum and Hagen. This article was published in EUUG Volume 3 nr 1, pp 12-13, Spring 1983[14].

```
/* Test 1 – CPUmemory */
main()
{ TYPE i, j, k;
  for(i = 0; i < 1000; i++)
       for(j = 0; j < 1000; j++)
            k = i + j + 1983;
}
```

Their design goal was to challenge the audience to measure performance on the many minis and micros that showed up at vendor exhibitions at UNIX Conferences.

Test 1 is for CPU/memory only and is a small program one can easily enter manually. Note, that small programs fit in cache memories. Also note that an optimizing compiler, that spots that variable ***k*** isn't used at all, isn't commonly used.

Test 2 is omitted here: it is about I/O performance.

To compare results of machines that are long gone with two systems (MicroVAX 3800 and PowerNode 6040) in my collection, I will pick the highest ranking commonly known systems from the long list in the paper:

elapsed usr secs system	short	register short	int	register int	long	register long
Concept 32/87	1.5	1.5	0.9	1.4	1.0	1.4
MicroVAX 3800	2.0	1.4	1.6	1.3	1.6	1.3
PowerNode 6040	4.0	4.0	4.1	2.0	4.1	2.0
VAX 11/780	8.8	8.6	6.7	4.7	6.7	4.7
Concept 32/27	12.0	11.0	10.0	10.0	10.0	10.0
VAX 11/750	16.2	16.5	10.9	7.3	10.8	7.3
MicroVAX	28.2	28.3	22.2	12.4	22.3	12.4
VAX11/730	37.4	37.5	23.9	14.4	24.1	14.4

- The Concept 32/87 is the ECL based system that wins a silver medal crossing the finish line after an Amdahl mainframe (see original paper).
- The MicroVAX 3800 did not exist in 1983 as it is on the market since 1989. Sales people said it be 3.8 times a VAX 11/780. This machine shows that MicroVAXen, built using LSI/VLSI technology, have grown up after all.
- In 1983 the PowerNode 6040 wasn't there either, but its sistership, the Concept 32/67, was.
- The Concept 32/27 is a low end machine with a single board CPU.

When comparing the machines in the table, you would like to know their price tag, as its all about price/performance ratio. We have to do without. The SEL-32 series successfully outperformed its competitors. *The myth of fire breathing dragons is confirmed.*

14 To be found at https://www.tuhs.org/Archive/Documentation/AUUGN/AUUGN-V05.4.pdf (p57 in pdf)

IT heritage and the simulation of historical computers

SIMH is about simulation of legacy computers on modern computers and reliving the old operating systems of long gone systems. The project was started in connection with the Computer History Museum in Boston. You will find simulators for a large range of old DEC, IBM, HP, DG and many other brands in the SIMH archives.

Building a simulator for an ancient history computer is not that simple. You need detailed documentation about both hardware and software, and lots of old software including diagnostics and last but not least the original operating systems. The chances to find this old material depends on the popularity and possible hobby-use of these computers.

Gould PowerNodes vanished around 1995. The bigger ECL based models are not suitable to be kept for hobby, due to the huge amount of power they use. Very few collectors do have leftover machines, but most have been scrapped a long time ago. For the SEL-32 systems, software and manuals about the technical details are hard to find more than 25 years after the last systems were scrapped. Manuals lose their significance to be preserved if there are no systems anymore.

Apart from the manuals, detailed knowledge is needed. If there is one person, who has climbed down to the deepest levels of the SEL-32 systems, it is Jim Bevier. Jim has worked for Systems Engineering Laboratories from 1976 until 1983. Later, working for another company, he has done more MPX/32 system software. In 2012 Jim wanted to run a few saved programs on a Linux machine at home and started developing an emulator and several cross tools for Linux. After his retirement in 2013 work continued.

Some of SIMH/sel32 development milestones:

- December 19, 2018: First release of Jim's SIMH/sel32 simulator merged into SIMH repository at github. It ran MPX/32 1.5F.
- May 19, 2019: my first contact with Jim. The tape (½" open reel) with diagnostics converted to SIMH .tap format and transported to Jim's desk in Arizona at the speed of light. Scanned documentation and distribution tapes for UTX/32 2.1A and 2.1B followed later.
- Jim worked through the set of CPU diagnostics verifying and debugging his simulator.
- Jan 14, 2020: Jim reports UTX/32 2.1A distribution tape (1 of 3) booting up to the # prompt. UTX runs using a memory disk and tape access only. Disks not yet working.
-

Disks are not that simple! Storage Module Drive (SMD) is an interface defined by Magnetic Peripherals Inc and provides a.o. controller dependant sector length.
SMD disks for Gould have one or two blocks spare per track as opposed to multiple blocks per disk to replace bad blocks. So we need book-keeping per track (track labels in addition to sector labels).

This is what the disk utility "prep" arranges before you install UTX. Disk preparation is a major step in the installation of UTX. It's about formatting and partition sizes but also about the bookkeeping of the disk in a.o. tracklabels.

Gould model 8887 (Fujitsu M2333K) 8" SMD disk, size 340MB

Tracklabels had to be supported by the simulator too and it turned out that the file used as a disk image under SIMH/sel32 is not just the payload of data but includes the book-keeping that UTX wants to see. If we don't satisfy "prep" we don't get the operating system installed!

Unlike as in SIMH/pdp11, you cannot copy a SIMH/sel32 disk-file one-to-one onto a real disk and kick off the operating system. Well, who keeps a PowerNode for hobby anyway? I do.

- Jan 25, 2020: the benchmark "zeef" compiled at the real PowerNode and put on a dump tape, loaded into the memory disk on the simulator after booting up from the boottape. Zeef is discussed later in this paper.
- July 2020: Jim managed to recreate an MPX/32 3.4 SDT tape[15] from bits and pieces and boot and run it.
- */usr/*games/rain used as stress test for output on console terminal.
- August 2020: simulation of SCSI disks works and runs UTX/32 2.1B.
- By end of august 2020 the network was up and running and a round trip from simulated sel32 to the real machine could be made by telnetting up and down.
- November 2020: the file UTX32-v21B-Binary-Kit.tgz was uploaded to bitsavers.org. The kit includes install and operations manual.
- June 2021: install MPX/32 3.6 from a hand built master SDT.
- March 2022: SIMH/sel32 officially released to run MPX/32 and UTX/32.

Benchmarking using ZEEF

"Zeef", the Sieve of Erathostenes, is a benchmark that has a static table holding the first 10,000 prime numbers and using that table will calculate the next 4,000 primes.

Processor	*native/ simulated*	*seconds elapsed*	*rating*
Gould PN 6040	native	1468	1
Raspberry PI 3B+	SIMH/sel32	461	3.18
IBM xSeries 345	SIMH/sel32	213	6.89
Raspberry PI 4B	SIMH/sel32	186	7.89
Odroid N2+	SIMH/sel32	133	11.04
AMD FX-6300	SIMH/sel32	50	29.36

It takes 36,384,896 calculations to find the 4,000 primes. Primes are integer, but checking the remainder of a division takes real numbers. In SIMH/sel32 floating point is emulated in software as the format of a floating point number in SEL/32 systems does not match the IEEE 754 standard used in modern CPUs. My real PowerNode 6040 does not have the optional FP hardware either.

The implementation of the IPU.

I proposed the idea of implementing the IPU using fork(), mmap() for the shared memory and semaphores. The basic idea was to have a second SIMH/sel32 forked just before it attached its

15 MPX/32 System Distribution Tape: either a distribution tape provided by Gould/SEL or a user generated bootable tape.

peripherals. That way, SIMH/sel32 was used as a blackbox to run the IPU.

- September 2022: the IPU challenge taken. I made a start with the experimental IPU implementation just to see how far I could get. I assumed UTX contained the support for the IPU, which proved true. The second SIMH/sel32 instance was forked off before the peripherals were attached and initialised. Using signals (SIGUSR1 and 2) to get some logging info, as the IPU does not have a console terminal attached.
- A small data structure is shared between the two processes to transfer the SIPU trap from CPU to IPU and in the other direction from IPU to CPU. Like interrupts from network or console terminal these traps are asynchronous; not expected and not waited for as result of an I/O operation.
- About ten I/O instructions had to be changed to give an “IPU undefined instruction” trap.
- For me there was still a big puzzle in questions like “*How to start the IPU and how to let it wait for the CPU?*”. The WAIT instruction seems most likely be used for waiting, but the cold started IPU hopped into a HALT instruction every time I tried. Time to contact Jim and ask for help.
- Jim restructured my attempt to implement the IPU and he shed a lot of light on some misunderstandings: HALT still listens for SIPU traps, a detail that I wasn’t aware of and obviously wasn’t implemented for the original single-cpu version of the simulator.
- Using fork to get a second process that simulates the IPU does not work on Microsoft Windows that is a platform for SIMH simulators too. Posix threading has been used as alternative. Chosing between the forked and threaded version is possible at source level.
- The instructions Set Bit in Memory (SBM) and Zero Bit in Memory (ZBM)[16] are used by the operating system to implement semaphores between CPU and IPU. At the SIMH/sel32 simulator level, the simulation of SBM and ZBM must be protected from interference when running simultaneously on CPU and IPU. The implementation of SBM and ZBM uses Posix Semaphores in the forked version and Mutex in the threaded version.

The experimental implementation of the IPU shows double or close to double performance on modern multicore processors. The forked version shows equal performance of CPU and IPU. The threaded version shows more difference in performance: the CPU is about 7 to 8% slower than the IPU. Older implementations of the POSIX threading function may perform differently.[17]

Elapsed time of zeef

threaded	forked
single: 125.8 s	single: 133.9 s
dual: 137.5 s / 127.5 s	dual: 134.0 s / 133.9 s

In the threaded version the IPU is significantly faster. In that particular case, the IPU does not check for SIMH events that may have happened. These events are (simulated) interrupts from channels that are attached to the CPU only. In the forked version both CPU and IPU perform about equal. Both check for events which only can happen at the CPU.

In the real PowerNodes and Concepts, the CPU and the IPU each have their own cache memory, which may cause inconsistencies in the shared memory. Cache control registers allow the operating system to define a range of memory that the local cache ignores in order to keep that range of memory consistent between the processors. In SIMH/sel32 cache is not implemented, but cache control is, albeit as a stub.

- Q1/2023: After debugging and testing, the implementation of the IPU is finished: the

16 SBM is named bsetw and ZBM bclrw in the assembler for UTX.

17 Use *getconf GNU_LIBPTHREAD_VERSION* to see NPTL version. At Odroid-n2+ (5.19) this is 2.35

experimental fork/mmap/semaphores version is complemented by a pthread/mutex version that runs on MS Windows. *The return of the fire breathing dragons is completed.*

UTX/32S C2 Secure UNIX

Spinoff from standard UTX/32 is a C2 Secure version named UTX/32S. This Unix version is pretty closed. Users work in a restricted environment and cannot see e.g. */dev.* System calls are passed onto the lower layer that consists of the real kernel. I have the tapes and docs, but UTX/32S isn't a neat toy-for-joy. It may appear on bitsavers.org one day.

UTX/32S is certified for use by U.S. Government and has been used in the military mainly. When Gould Electronics was sold to Nippon Mining the installed base of government related systems was the reason to keep the Computer Systems Division in U.S. hands and it was sold off to Encore Computers.

The NPL systems.

The New Product Line (NPL) intended to succeed the Concept Product Line (CPL).

The NP1 ran UTX/32 v3 and could address up to 4 GB of physical memory extending the limitations of the 24 bits addresses (max 16MB) used in the CPL models. Its main system bus could transfer up to 150 MB/sec. Development started in 1983 and the first units shipped in 1987.

The NP1 models ranged in price from $395,000 to $2.9 million. The largest system, the Model 480, has up to 4 GB bytes of physical memory. About 100 units were sold but the next model, the NP2, didn't make it to the market.

See:
https://www.paralogos.com/DeadSuper/Misc/Gould.html

Later ENCORE systems

Early 1990s the Encore 91 (and 93) was announced. This was a symmetric multiprocessor system based on Motorola 88100 processors and ran Encore UMAX V with optional MicroMPX extensions to support MPX-32 compatible functions.

RETROSPECTIVE

By the end of the 80s the Concept Product Line (CPL) ran against its limitations of max physical memory of 16 MB. It sufficed using serial terminals but those terminal were getting out of fashion.

New developments in UNIX that came after the latest UTX/32:

- Shared libraries and dynamically linked objects. Implementations for System V release 4 presented at USENIX conference in 1986 (Atlanta) and for SunOS 4.x in 1987 (Phoenix).
- Virtual I/O. Paging from/to a file. Copy a large file to /dev/null in the blink of an eye.
- Modern C standard. Current compiler still based on the Portable C compiler. For instance, GNU-C is not available, although there has existed a version of GDB.

Early 90s, X11 Windows by MIT, later the X Consortium, got very popular in scientific environments. The above mentioned limitations in hardware and software made it hard to implement and deploy. X11 Windows featured running graphical applications on the central host and use a display on your desk.

SIMH is about serializing all that happens in parallel in real hardware. Both in the forked and the ptheaded version, functions offered by the operating system are used that are not offered (by SIMH) to the sim developer. For that reason, the implementation of the IPU using SIMH as a layer of supporting software does not comply with the rules of the game.
In my humble opinion, the initial implementation using fork() fits closely with SIMH intentions as the CPU and the IPU are independant entities. Obviously, fork() is just used to get it started. As a result CPU and IPU operate as two SIMH simulators using just shared memory and semaphores. The result works great, when running on today's multi-core processors.

Acknowledgements

There is close to nothing on the Internet about the history of SEL and Gould CSD except for a lot scattered bits and pieces. Thanks to Jim and Wytze for highlighting a handful of black spots in this history. Thanks to Ronald for info on where systems were deployed in The Netherlands. Thanks to John, Wytze and Sean for reviewing this paper. Last but not least, thanks Jim for your hard work and for the fun I had in contributing and working with you.

Resources and Pointers

- SIMH/sel32 at Jim's Github Repository: https://github.com/AZBevier/sims
- Tapes, command files to testdrive your dragon: https://github.com/AZBevier/SEL32-installs
- Website by Geert : https://geerol.home.xs4all.nl/sel32.htm
- OpenSIMH Repository: https://github.com/open-simh/simh
- Resources at bitsavers.org:
  - software: https://bitsavers.org/bits/SEL/
  - docs: https://bitsavers.org/pdf/sel/

This document is released in September 2023.