USENIX Association

Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

USENIX
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

plex86: an i80x86 virtual machine

Kevin Lawton

MandrakeSoft, inc.

What is virtualization ?

- It is desirable to be able to run multiple operating systems simultaneously:
  - in order to simultaneously run application software written for different operating systems
  - in order to debug system software using a fully equipped host debugging environment
  - in order to consolidate management of a great number of heterogeneous PC environments in an enterprise to a small set of virtual machine servers.

What is virtualization ?

- One could attempt to emulate the complete hardware environment — BOCHS
  - Pro: works on any platform, emulators are portable
  - Con: this is very slow
- One could attempt to emulate the API of the different operating systems — WINE, UML
  - This requires the host platform to be the same as the platform of the emulated operating system
  - The emulated API needs to be fully documented, which is not the case for most commercial systems

What is virtualization ?

- The intermediate solution is virtualization — plex86
  - Emulate instructions in the guest OS which access features which can not be used concurrently with the host OS
    - The peripheral devices
    - System instructions
  - Create an environment which will allow all other instructions to be executed natively by the processor

How does virtualization work ?

- Execute all virtualized code in user-mode, and the virtual machine monitor in supervisor mode
- The processor will trap to the monitor when a privileged instruction is executed in user-mode; thus, we can emulate all privileged instructions in the monitor and let the rest run at full speed

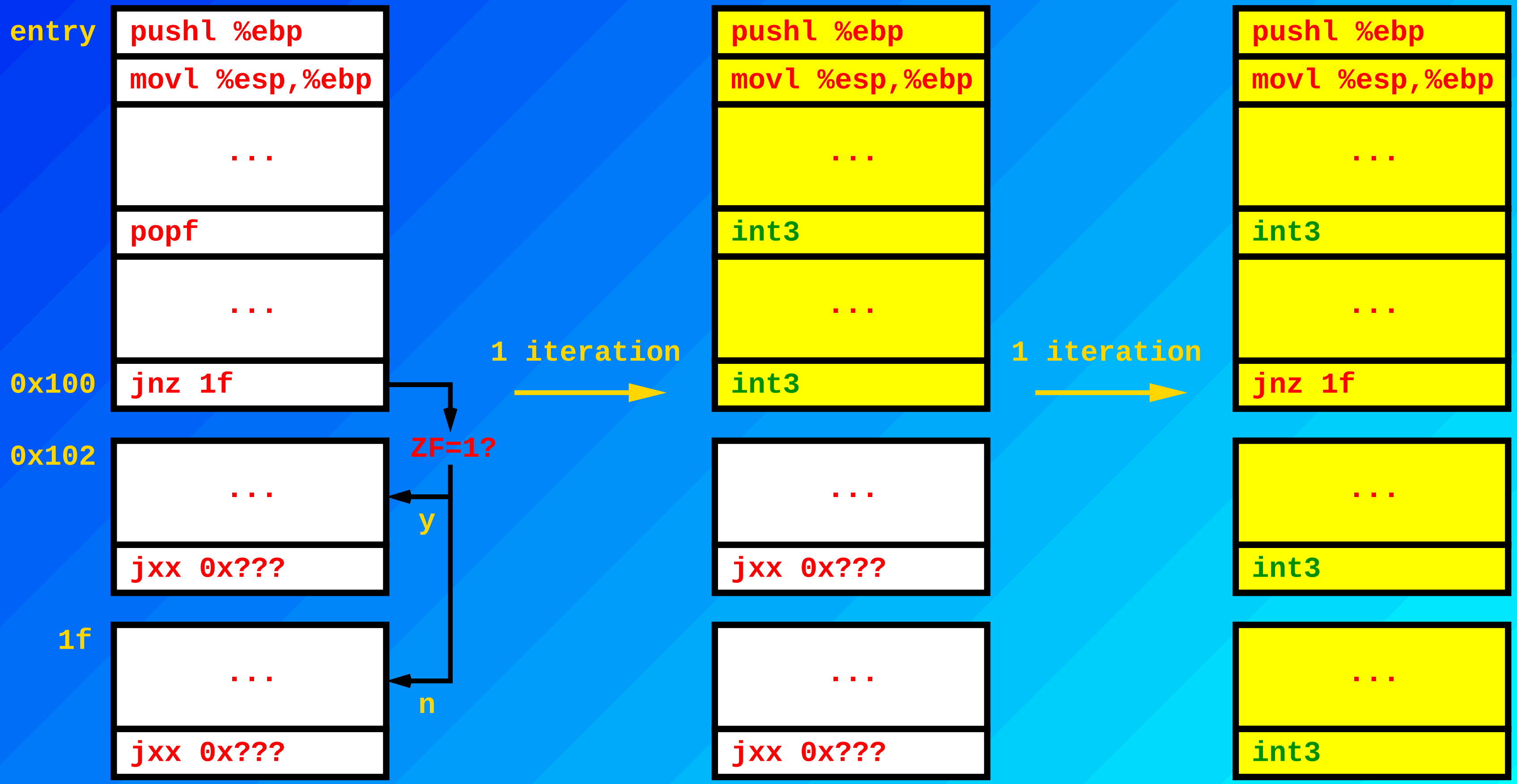
Challenge on the Intel 80x86

- The Intel 80x86 series were not designed with virtualization in mind:
  - There are instructions which need to be virtualized which are not privileged instructions: `sgdt`, `sidt`, `smsw`, `str`, etc.
  - There are instructions which behave differently in supervisor-mode (ring 0) than in user-mode (ring 3): `pushf`, `popf`.
- Challenge: find a way to make these instructions trap anyway

The plex86 solution: SBE

- The SBE = Scan Before Execute method aims to scan the user code before execution, replacing instructions to be virtualized with the trap opcode `int3`
- Problem: the Halting Problem of theoretical computer science: the execution of a program cannot be predicted in advance
- Solution: Dynamic SBE: scan “during execution”

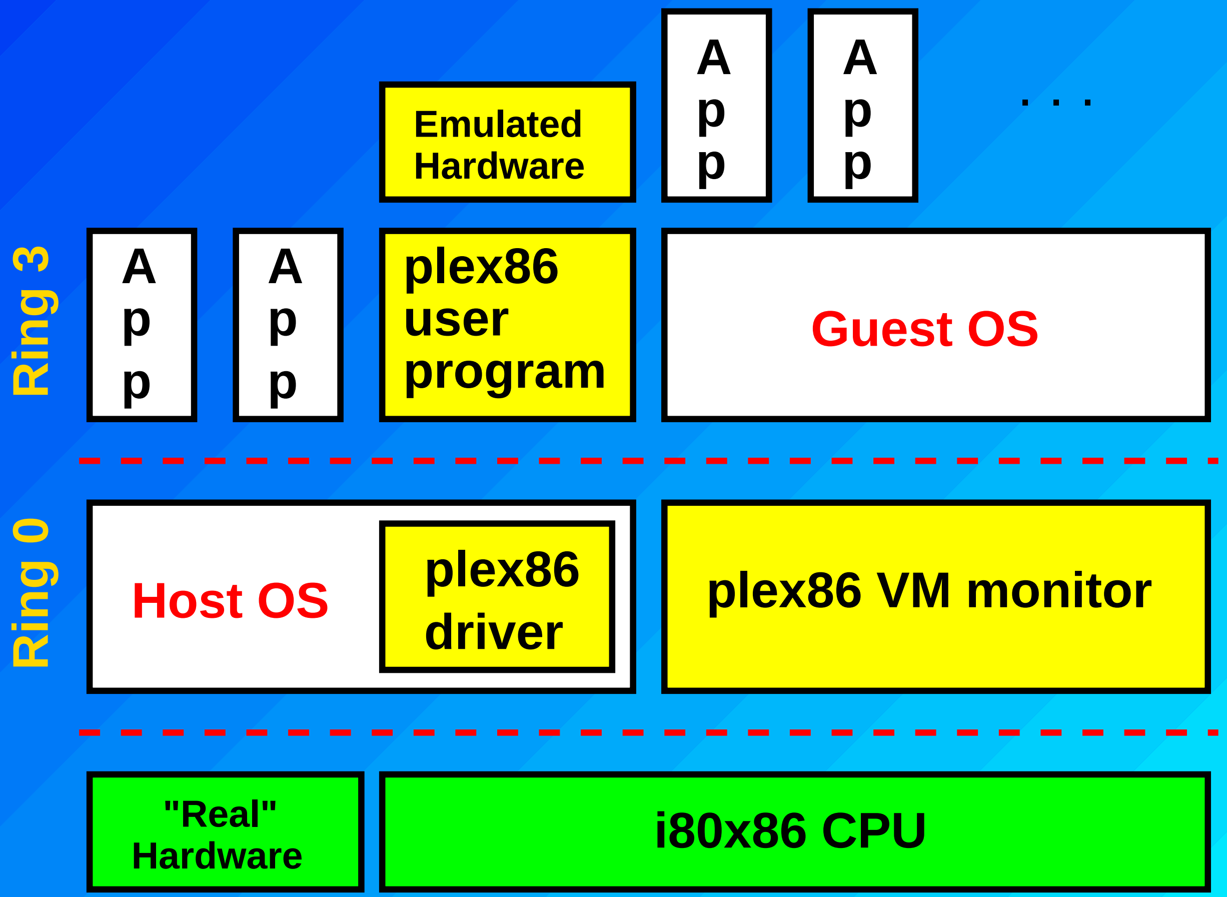
Implementation of Dynamic SBE



Implementation of Dynamic SBE

- For a scanned code page three separate pages are used:
  1. a copy of the original code page
  2. a scanned code page, which the processor executes from
  3. an attribute page, which contains information on which parts of the code page have been scanned and which instructions have been replaced (yellow and green colors on previous slide)
- Scanned pages are cached for performance reasons

plex86 architecture overview

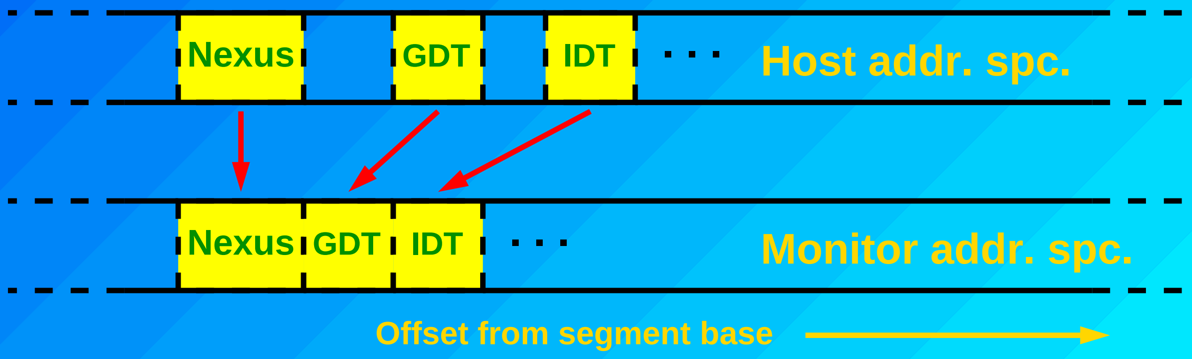


plex86 architecture overview

- The virtual machine has its own processor context, including its own GDT, IDT, and page tables
- The main job of the host OS driver (kernel module under linux) is to switch back and forth between the virtual machine's context and the host OS's context
- The virtual machine monitor is responsible for SBE, instruction emulation, and timing, amongst others
- The plex86 user program provides an interface to the perhiperal hardware emulation

plex86 architecture overview

- The monitor is written in Position Independent Code and is collected into one page directory in the guest address space: it can be freely moved around, so guest and monitor never use the same memory
- The context switching code (nexus) is mapped at the same logical address in monitor and host, and functions as a "wormhole" in the transition



plex86 architecture overview

- The user program itself is minimal: all of the “real” functionality is contained in dynamically-loaded plugins
  - Perhiperal hardware emulation is implemented in plugins (currently extracted from BOCHS)
  - Plugins can be used to extend the VM in many ways, for instance: the ICE plugin provides system debugging through GDB
- The plugin system makes plex86 easy to extend and maintain.

Plugins are easy!

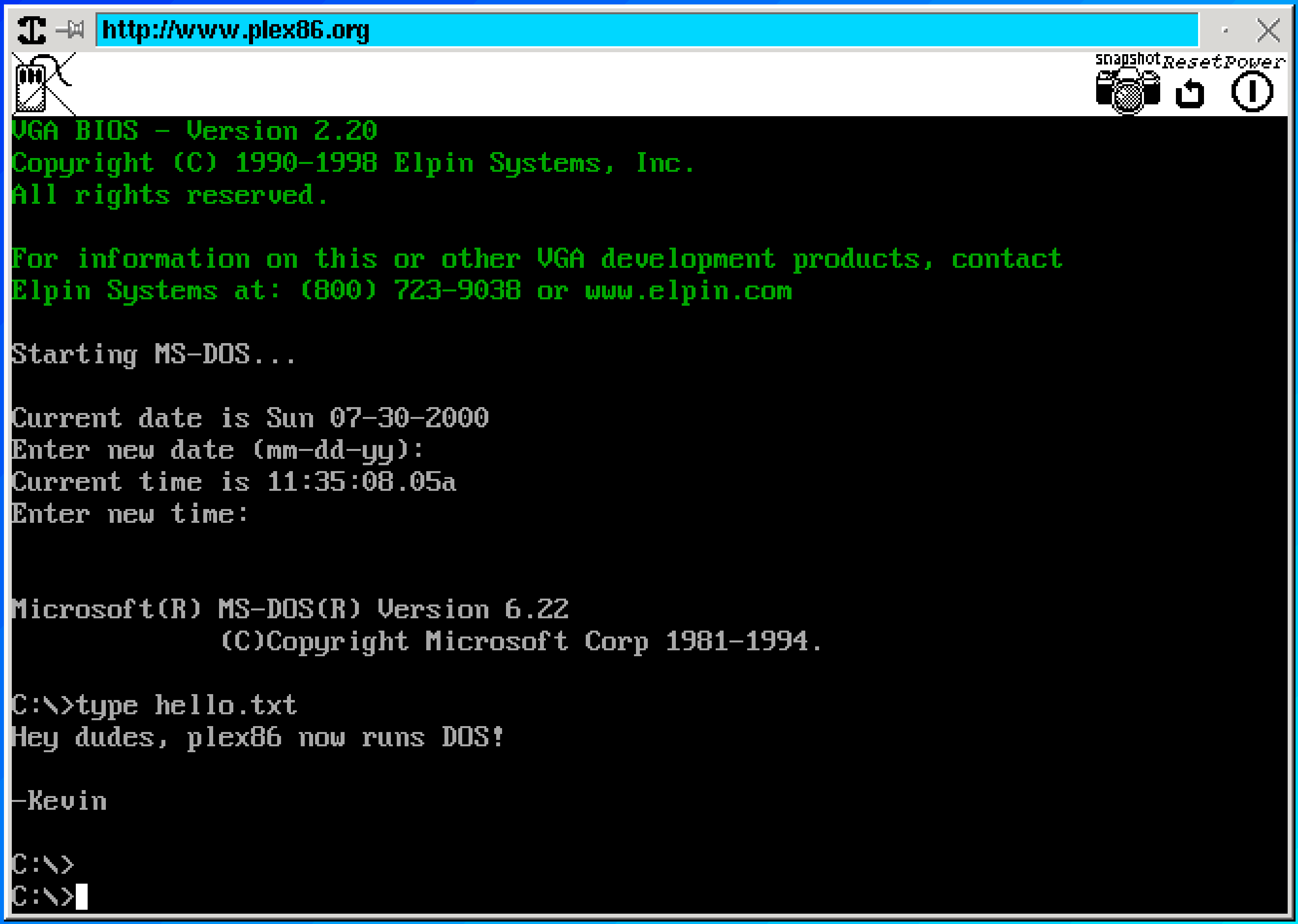
Writing a plugin requires little knowledge of plex86 internals, making it very accessible. A trivial example:

```
plugin_t *simple_plugin;
int plugin_init (plugin_t *plugin, int argc, char *argv[])
{
    simple_plugin = plugin;                             /* Store plugin handle          */
    plugin_alloc_outp (simple_plugin, io, 0x300, 1);    /* Allocate port 0x300 for output */
    return 0;
}
void plugin_fini (void)
{
    plugin_free_outp (simple_plugin, 0x300, 1);         /* Free port 0x300              */
}
int io (event_t evt, int data, int op_size, int count, void *loc)
{
    printf ("simple plugin got 0x%x on port 0x300\n", *(char *)loc);
    return 0;
}
```

Current status

- The main framework for all aspects of plex86 is done
- Many instructions are not yet emulated correctly
- SBE is yet unoptimized, and thus slow
- Several special guest test-kernels can be run without problems
- Microsoft MS-DOS and FreeDOS appear to work pretty well for textmode applications
- Linux crashes in the initialization sequence, but is rapidly getting further through it

Current status



```
VGA BIOS - Version 2.20
Copyright (C) 1990-1998 Elpin Systems, Inc.
All rights reserved.

For information on this or other VGA development products, contact
Elpin Systems at: (800) 723-9038 or www.elpin.com

Starting MS-DOS...

Current date is Sun 07-30-2000
Enter new date (mm-dd-yy):
Current time is 11:35:08.05a
Enter new time:


Microsoft(R) MS-DOS(R) Version 6.22
             (C)Copyright Microsoft Corp 1981-1994.

C:\>type hello.txt
Hey dudes, plex86 now runs DOS!

-Kevin

C:\>
C:\>
```

We're very proud of this screenshot!

Future directions

- A hardware interface that is efficient for real hardware is not necessarily efficient for virtualized hardware. Special plex86 hardware can be "developed" providing better performance for the guest. Special guest drivers need to be written to support the plex86 hardware
  - The guest OS could be made to communicate directly with the host OS graphics subsystem, thus eliminating the VGA emulation overhead. For portability, a library such as SDL will be used
  - Similar constructions can be used for speeding up networking, etc.

Future directions

- The virtual machine can be made persistent. The VM dumps the memory core and hardware state to a file, which can be reloaded at any time. This saves reboot time, keeps programs open, etc.
- Another useful option is persistent disk images. Changes caused by writes to a drive are ordinarily made directly to the disk image. However, one may enable a mode where changes are written to a “diff file” in stead. This is useful, when you just messed up a big install: just undo it!

Future directions

- We are considering adding Python scripting support to plex86. This can be useful for large debugging runs and the like.
- Virtual Machine Service Providers: There is interest by third parties to use plex86 on high-powered servers to offer VM services, accessible via the Internet to specific client locations (fixed) or from anywhere (mobile)

Future directions

- The possibilities of porting the plex86 framework to other architectures, in particular the Intel Itanium and AMD Hammer platforms, should be explored
- Got any more ideas? Let us know!

For more information:

www.plex86.org