Why pv_ops kernel?

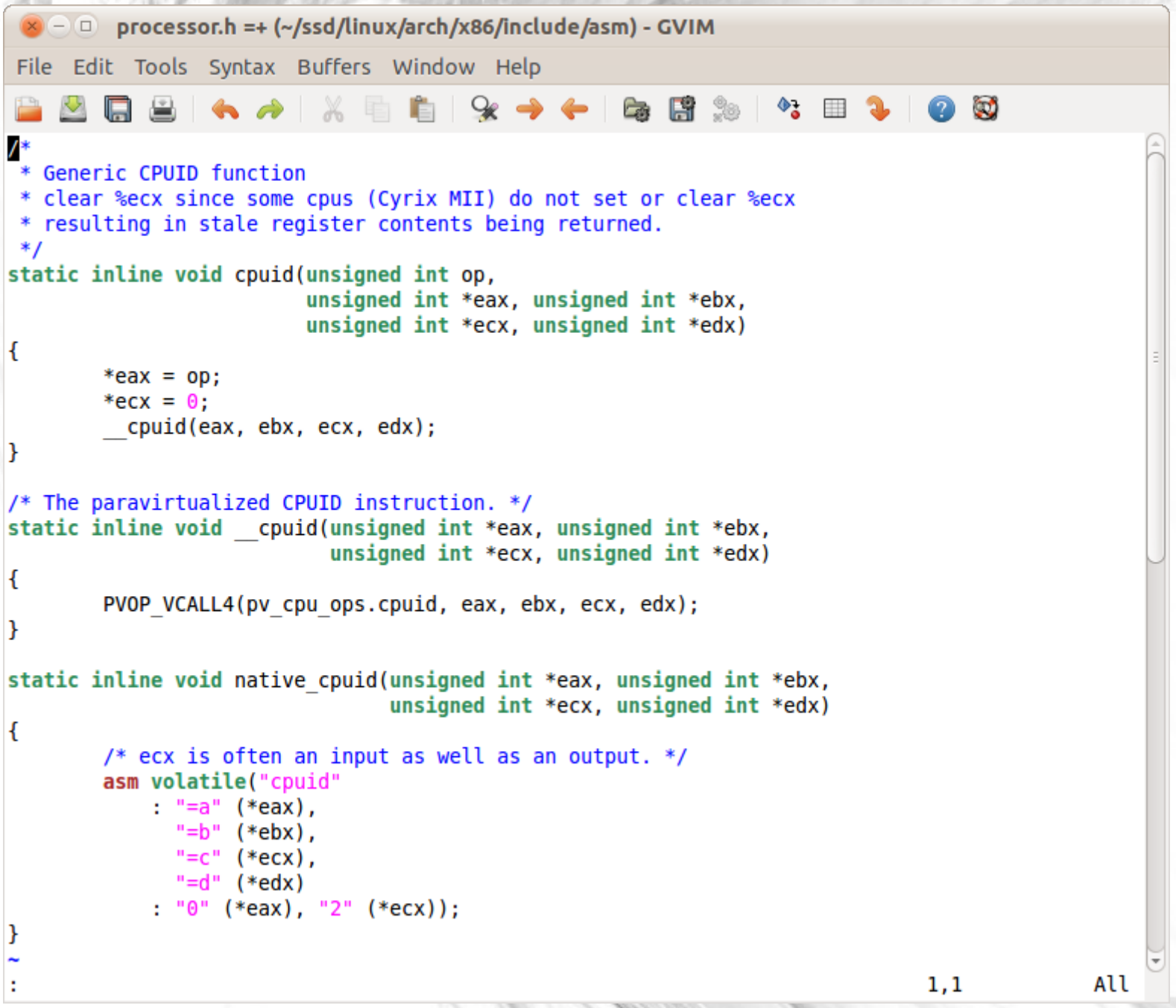
- Maintance headaches with Xen-O-Linux patches (common code in two sub-arches)
- Linux distros scuff at testing two kernels (baremetal and -xen)
- Not upstreamable

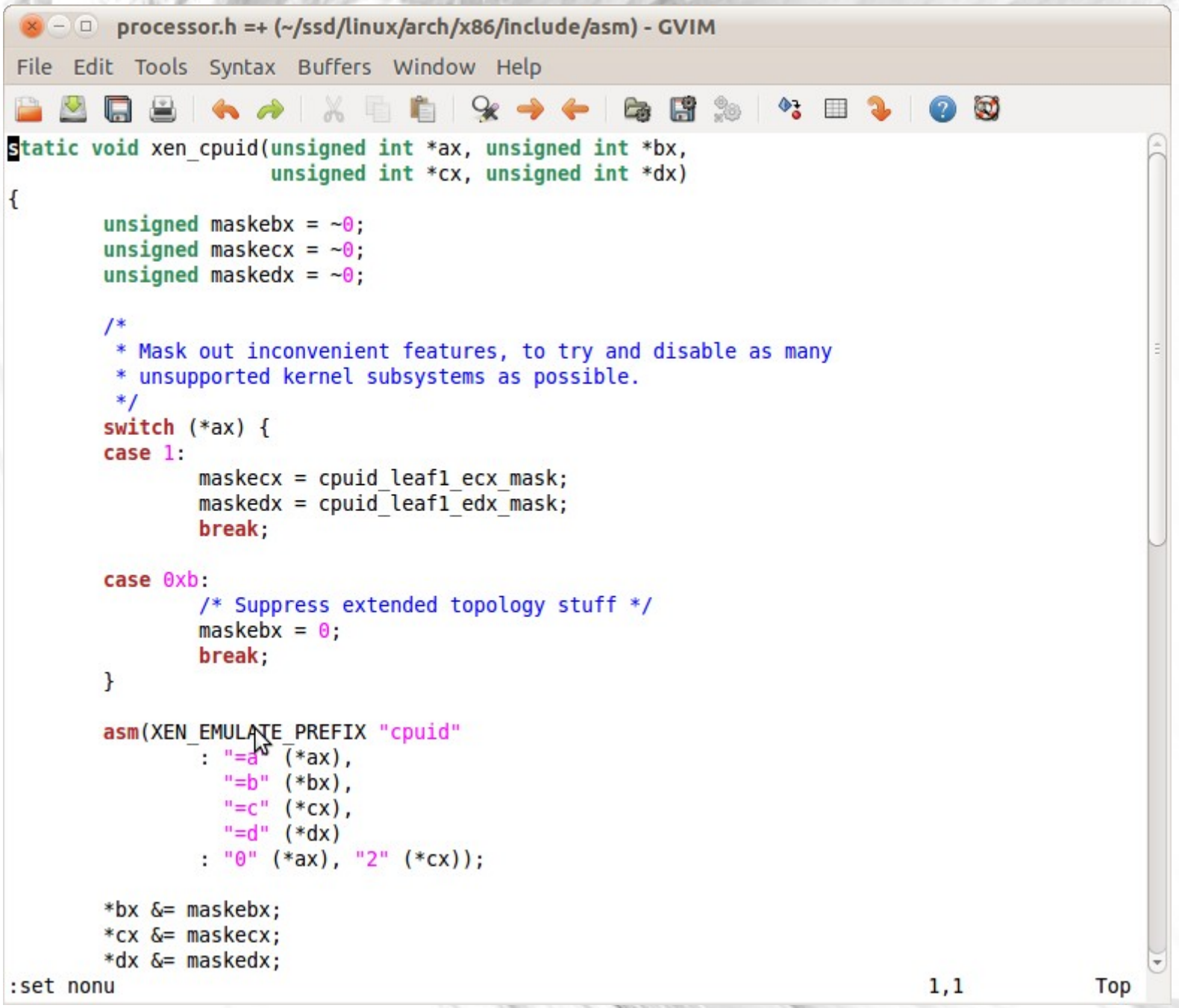
Requirements

- One kernel that can boot
  - On baremetal hardware
  - Virtualization friendly
  - Paravirtualization capable
- Existing frameworks could be used:
  - (alternative_asm) Identify if by cpuid whether SSE3 is on, and if so use those for memcpy.
  - SMP vs uni-processor – adds 'lock' or removes it. (smp_alt)

Ideas

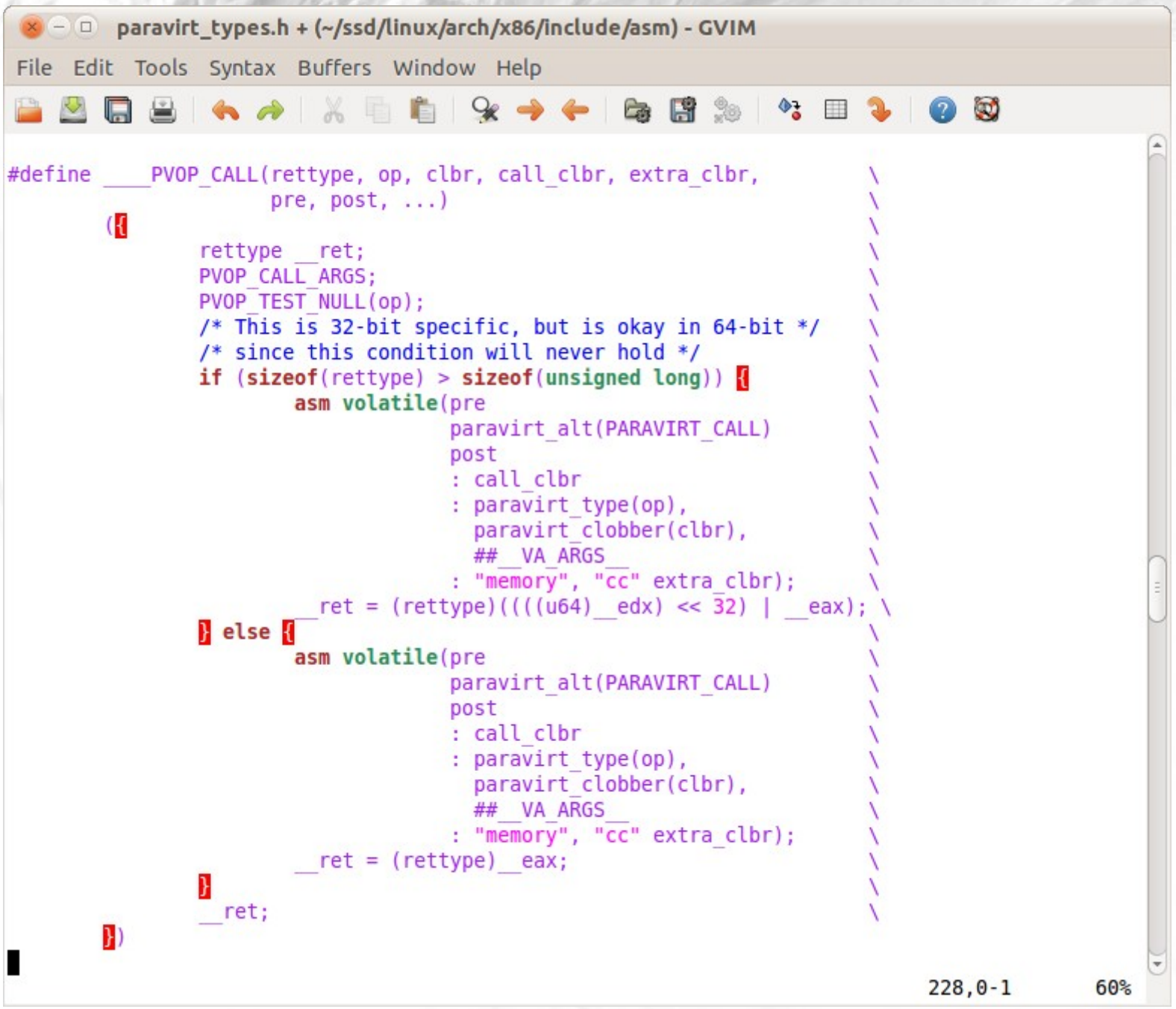
- Old code has:
  - #ifdef CONFIG_XEN
- Replace it with:
  - If (xen_domain()) ..
- Or altgother remove those and wrap calls:
  - cpuid(...) → pv_cpu_ops->cpuid(...) → native_cpuid(...)
  - arch/x86/include/asm/paravirt.h has the pv_cpu_ops defintions
  - arch/x86/kernel/paravirt.c has the native

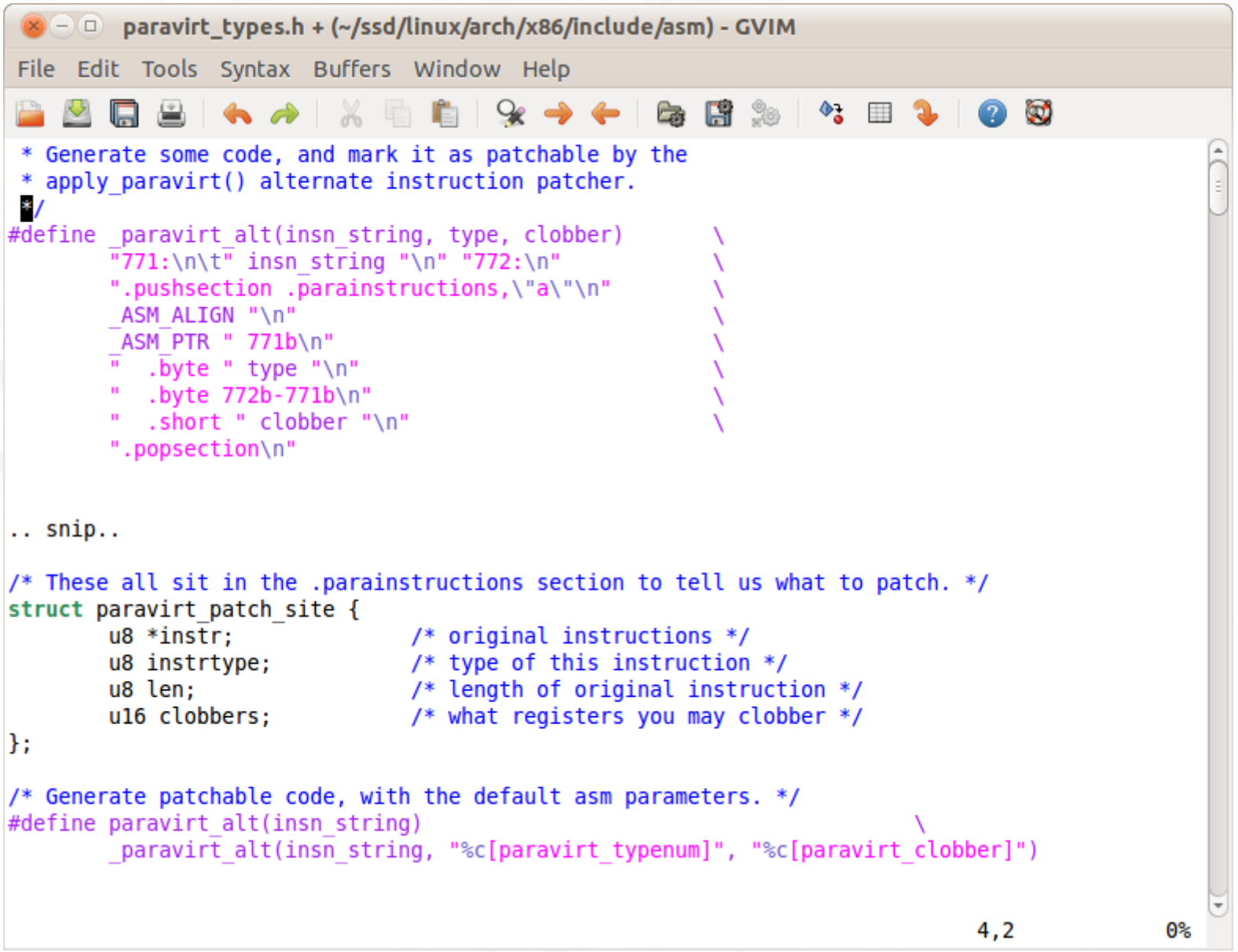

processor.h =+ (~/ssd/linux/arch/x86/include/asm) - GVIM
File Edit Tools Syntax Buffers Window Help
/*
* Generic CPUID function
* clear %ecx since some cpus (Cyrix MII) do not set or clear %ecx
* resulting in stale register contents being returned.
*/
static inline void cpuid(unsigned int op,
unsigned int *eax, unsigned int *ebx,
unsigned int *ecx, unsigned int *edx)
{
*eax = op;
*ecx = 0;
__cpuid(eax, ebx, ecx, edx);
}
/* The paravirtualized CPUID instruction. */
static inline void __cpuid(unsigned int *eax, unsigned int *ebx,
unsigned int *ecx, unsigned int *edx)
{
PVOP_VCALL4(pv_cpu_ops.cpuid, eax, ebx, ecx, edx);
}
static inline void native_cpuid(unsigned int *eax, unsigned int *ebx,
unsigned int *ecx, unsigned int *edx)
{
/* ecx is often an input as well as an output. */
asm volatile("cpuid"
: "=a" (*eax),
"=b" (*ebx),
"=c" (*ecx),
"=d" (*edx)
: "0" (*eax), "2" (*ecx));
}
~
:
1,1
All


processor.h =+ (~/ssd/linux/arch/x86/include/asm) - GVIM
File Edit Tools Syntax Buffers Window Help
static void xen_cpuid(unsigned int *ax, unsigned int *bx,
unsigned int *cx, unsigned int *dx)
{
unsigned maskebx = ~0;
unsigned maskecx = ~0;
unsigned maskedx = ~0;
/*
* Mask out inconvenient features, to try and disable as many
* unsupported kernel subsystems as possible.
*/
switch (*ax) {
case 1:
maskecx = cpuid_leaf1_ecx_mask;
maskedx = cpuid_leaf1_edx_mask;
break;
case 0xb:
/* Suppress extended topology stuff */
maskebx = 0;
break;
}
asm(XEN_EMULATE_PREFIX "cpuid"
: "=a" (*ax),
"=b" (*bx),
"=c" (*cx),
"=d" (*dx)
: "0" (*ax), "2" (*cx));
*bx &= maskebx;
*cx &= maskecx;
*dx &= maskedx;
:set nonu
1,1
Top

Not too fast..

- Binary patching to optimize path
- We want cpuid() to be replaced with native_cpuid() (or xen_cpuid)
  - cpuid (..)→ native_cpuid(..)
  - Perhaps replace cpuid(..) with native_cpuid(..) ?
- We need to know where in the kernel and with what call to replace it with (offset).
- Remeber that PVOPS_VCALL4?


paravirt_types.h + (~/ssd/linux/arch/x86/include/asm) - GVIM
File Edit Tools Syntax Buffers Window Help
#define ____PVOP_CALL(rettype, op, clbr, call_clbr, extra_clbr, \
pre, post, ...) \
({ \
rettype __ret; \
PVOP_CALL_ARGS; \
PVOP_TEST_NULL(op); \
/* This is 32-bit specific, but is okay in 64-bit */ \
/* since this condition will never hold */ \
if (sizeof(rettype) > sizeof(unsigned long)) { \
asm volatile(pre \
paravirt_alt(PARAVIRT_CALL) \
post \
: call_clbr \
: paravirt_type(op), \
paravirt_clobber(clbr), \
##__VA_ARGS__ \
: "memory", "cc" extra_clbr); \
__ret = (rettype)((((u64)__edx) << 32) | __eax); \
} else { \
asm volatile(pre \
paravirt_alt(PARAVIRT_CALL) \
post \
: call_clbr \
: paravirt_type(op), \
paravirt_clobber(clbr), \
##__VA_ARGS__ \
: "memory", "cc" extra_clbr); \
__ret = (rettype)__eax; \
} \
__ret; \
})
228,0-1 60%


paravirt_types.h + (~/ssd/linux/arch/x86/include/asm) - GVIM
File Edit Tools Syntax Buffers Window Help
* Generate some code, and mark it as patchable by the
* apply_paravirt() alternate instruction patcher.
*/
#define _paravirt_alt(insn_string, type, clobber) \
"771:\n\t" insn_string "\n" "772:\n" \
".pushsection .parainstructions,\"a\"\n" \
_ASM_ALIGN "\n" \
_ASM_PTR " 771b\n" \
" .byte " type "\n" \
" .byte 772b-771b\n" \
" .short " clobber "\n" \
".popsection\n"
.. snip..
/* These all sit in the .parainstructions section to tell us what to patch. */
struct paravirt_patch_site {
u8 *instr; /* original instructions */
u8 instrtype; /* type of this instruction */
u8 len; /* length of original instruction */
u16 clobbers; /* what registers you may clobber */
};
/* Generate patchable code, with the default asm parameters. */
#define paravirt_alt(insn_string) \
_paravirt_alt(insn_string, "%c[paravirt_typenum]", "%c[paravirt_clobber]")
4,2 0%

paravirt_types.h + (~/ssd/linux/arch/x86/include/asm) - GVIM

File Edit Tools Syntax Buffers Window Help

```
static void __cpuinit init_centaur(struct cpuinfo_x86 *c)
{
.. snip..
        switch (c->x86) {
        case 5:
... snip..
                if (cpuid_eax(0x80000000) >= 0x80000005) {
                        /* Yes, we can. */

In assembler:

        .section        .cpuinit.text
        .type   init_centaur, @function
init_centaur:
.LFB1017:
        .loc 1 341 0
        .cfi_startproc
.LVL2:
        pushq   %rbp    #

... snip ..
.LCFI4:
        .cfi_def_cfa_offset 16
        .loc 3 31 0
        movq    %r14, %rdi      # eax.93,
        movq    %r13, %rsi      # ebx.94,
        movq    %r12, %rdx      # ecx.95,
        movq    %r15, %rcx      # edx.96,
#APP
# 31 "/home/konrad/ssd/linux/arch/x86/include/asm/paravirt.h" 1
        771:
        call *pv_cpu_ops+232;   #
772:
.pushsection .parainstructions,"a"
 .balign 8
 .quad  771b
  .byte 31      #
  .byte 772b-771b
  .short 511    #
.popsection
~
~
:1                                                      1,1           All
```

In .text section init_centuar @**ffffffff81432725** this code:

- ff 14 25 90 01 82 81
- callq *0xffffffff81820190

In .parainstructions section @ line 443 ffffffff81921c40 this blob:

- **25274381 ffffffff**
- 1f07ff01 00000000
- %'C.....

.parainstructions:

443 ffffffff81921c40 **25274381 ffffffff** 1f07ff01 00000000 %'C.....

struct paravirt_patch_site {

u8 *instr; /* original instructions */ [ffffffff81432725]

u8 instrtype; /* type of this instruction */ [0x1f = 32]

u8 len; /* length of original instruction */ [0x07 = 8 bytes]

u16 clobbers; /* what registers you may clobber */ [0xff = 511]

};

In init_centuar:

ffffffff81432725: ff 14 25 90 01 82 81 callq *0xffffffff81820190

In pv_cpu_ops:

ffffffff8182018b: 81 ff ff ff ff **9a** cmp $0x9affffff,%edi

ffffffff81820191: **84 02** test %al,(%rdx)

ffffffff81820193: **81 ff ff ff ff** 56 cmp $0x56ffffff,%edi

ffffffff8102849a <native_cpuid>:

We overwrite @ffffffff81921c40 with **ffffffff8102849a**

Binary patching under Xen

In init_centuar:

ffffffff81432725: ff 14 25 90 01 82 81 callq *0xffffffff81820190

In pv_cpu_ops:

ffffffff8182018b: 81 ff ff ff ff **9a** cmp $0x9affffff,%edi

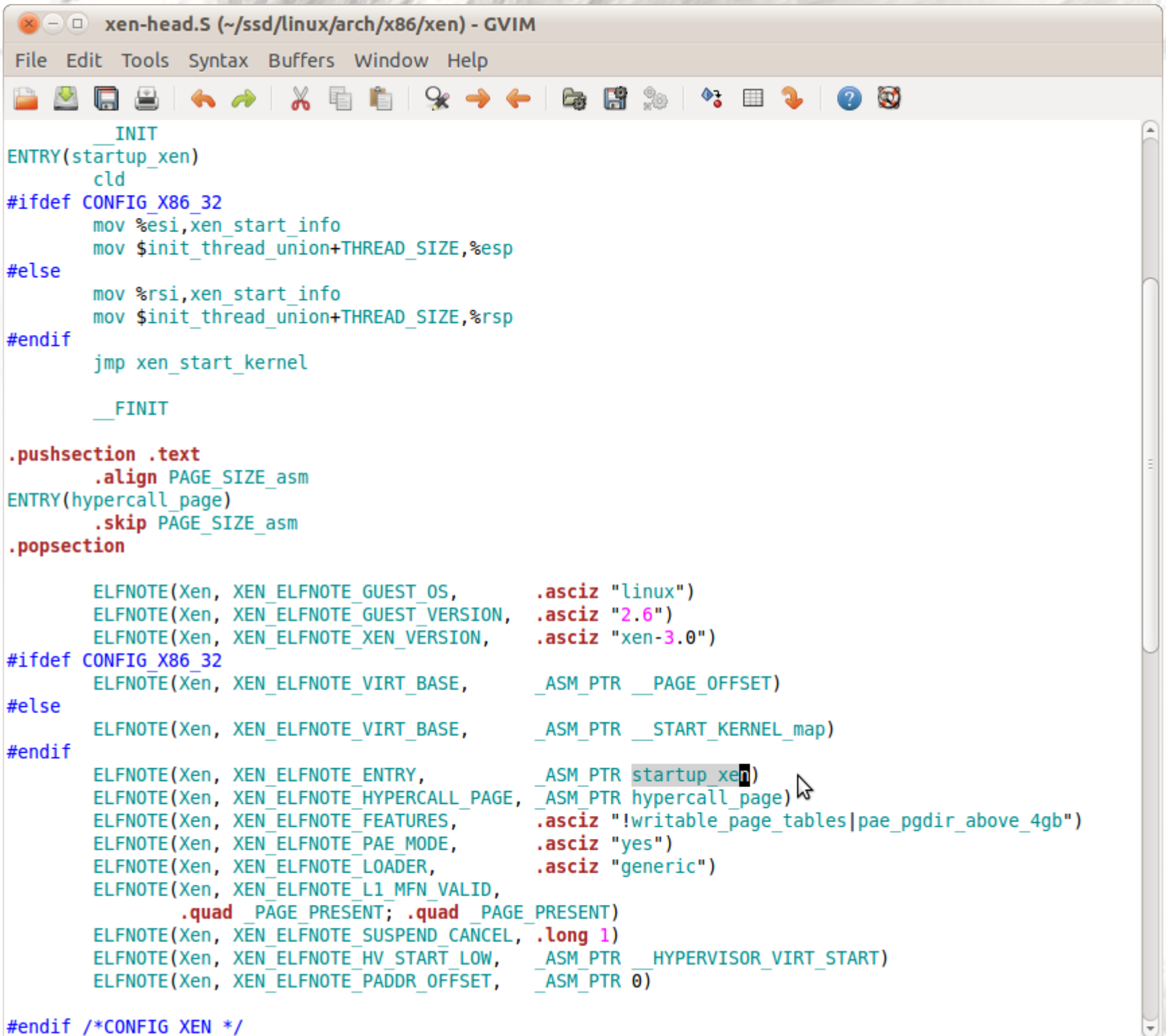
ffffffff81820191: **84 02** test %al,(%rdx)

ffffffff81820193: **81 ff ff ff ff** 56 cmp $0x56ffffff,%edi

under Xen, the pv_cpu_ops would be over-written with another struct.

ffffffff81003062 <xen_cpuid>:

native_patch calls paravirt_patch_insn, which figures out the delta and makes it a call ffffffff81003062


xen-head.S (~/ssd/linux/arch/x86/xen) - GVIM
File Edit Tools Syntax Buffers Window Help
__INIT
ENTRY(startup_xen)
cld
#ifdef CONFIG_X86_32
mov %esi,xen_start_info
mov $init_thread_union+THREAD_SIZE,%esp
#else
mov %rsi,xen_start_info
mov $init_thread_union+THREAD_SIZE,%rsp
#endif
jmp xen_start_kernel
__FINIT
.pushsection .text
.align PAGE_SIZE_asm
ENTRY(hypercall_page)
.skip PAGE_SIZE_asm
.popsection
ELFNOTE(Xen, XEN_ELFNOTE_GUEST_OS, .asciz "linux")
ELFNOTE(Xen, XEN_ELFNOTE_GUEST_VERSION, .asciz "2.6")
ELFNOTE(Xen, XEN_ELFNOTE_XEN_VERSION, .asciz "xen-3.0")
#ifdef CONFIG_X86_32
ELFNOTE(Xen, XEN_ELFNOTE_VIRT_BASE, _ASM_PTR __PAGE_OFFSET)
#else
ELFNOTE(Xen, XEN_ELFNOTE_VIRT_BASE, _ASM_PTR __START_KERNEL_map)
#endif
ELFNOTE(Xen, XEN_ELFNOTE_ENTRY, _ASM_PTR startup_xen)
ELFNOTE(Xen, XEN_ELFNOTE_HYPERCALL_PAGE, _ASM_PTR hypercall_page)
ELFNOTE(Xen, XEN_ELFNOTE_FEATURES, .asciz "!writable_page_tables|pae_pgdir_above_4gb")
ELFNOTE(Xen, XEN_ELFNOTE_PAE_MODE, .asciz "yes")
ELFNOTE(Xen, XEN_ELFNOTE_LOADER, .asciz "generic")
ELFNOTE(Xen, XEN_ELFNOTE_L1_MFN_VALID,
.quad _PAGE_PRESENT; .quad _PAGE_PRESENT)
ELFNOTE(Xen, XEN_ELFNOTE_SUSPEND_CANCEL, .long 1)
ELFNOTE(Xen, XEN_ELFNOTE_HV_START_LOW, _ASM_PTR __HYPERVISOR_VIRT_START)
ELFNOTE(Xen, XEN_ELFNOTE_PADDR_OFFSET, _ASM_PTR 0)
#endif /*CONFIG_XEN */

pvops

- Has two differences from ”classic” Xen:
  - pv_cpu_ops->func(..)
  - Binary patching