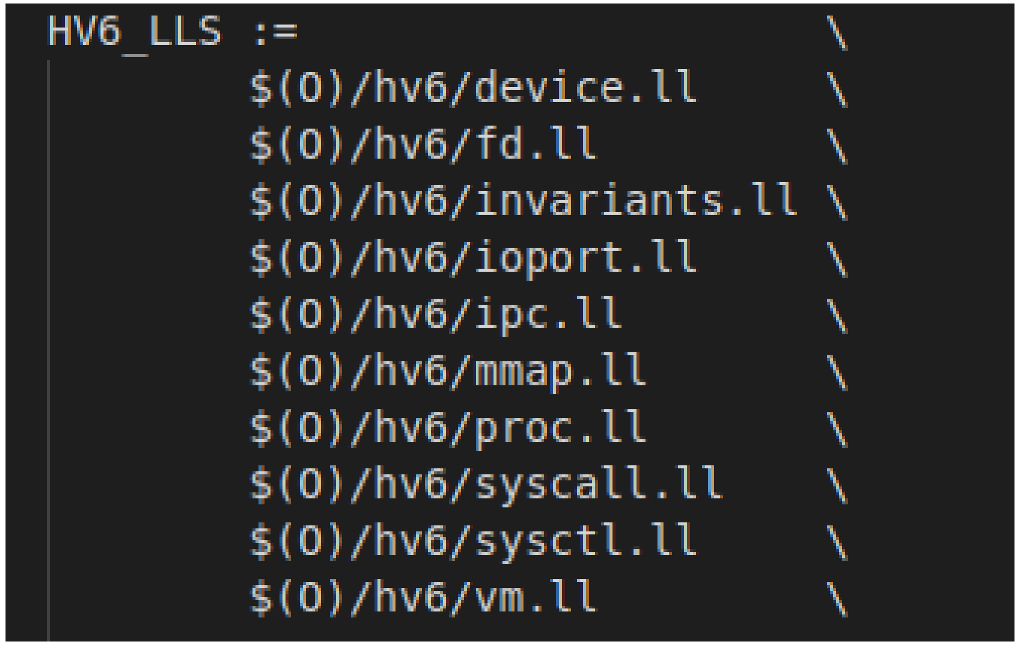
HyperKernel在RISC-V的移植与CPU结合验证 (g09)

秦岳 2015011333

陈宇 2015011343

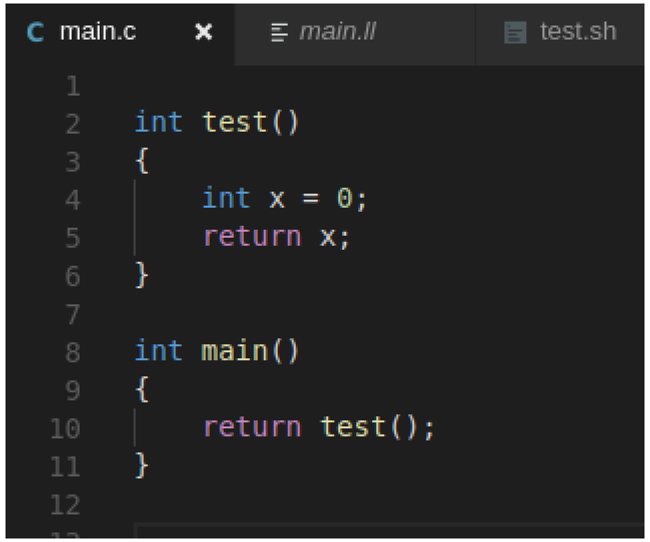
验证流程



```
HV6_LLS :=                          \
        $(O)/hv6/device.ll          \
        $(O)/hv6/fd.ll              \
        $(O)/hv6/invariants.ll      \
        $(O)/hv6/ioport.ll          \
        $(O)/hv6/ipc.ll             \
        $(O)/hv6/mmap.ll            \
        $(O)/hv6/proc.ll            \
        $(O)/hv6/syscall.ll         \
        $(O)/hv6/sysctl.ll          \
        $(O)/hv6/vm.ll              \
```

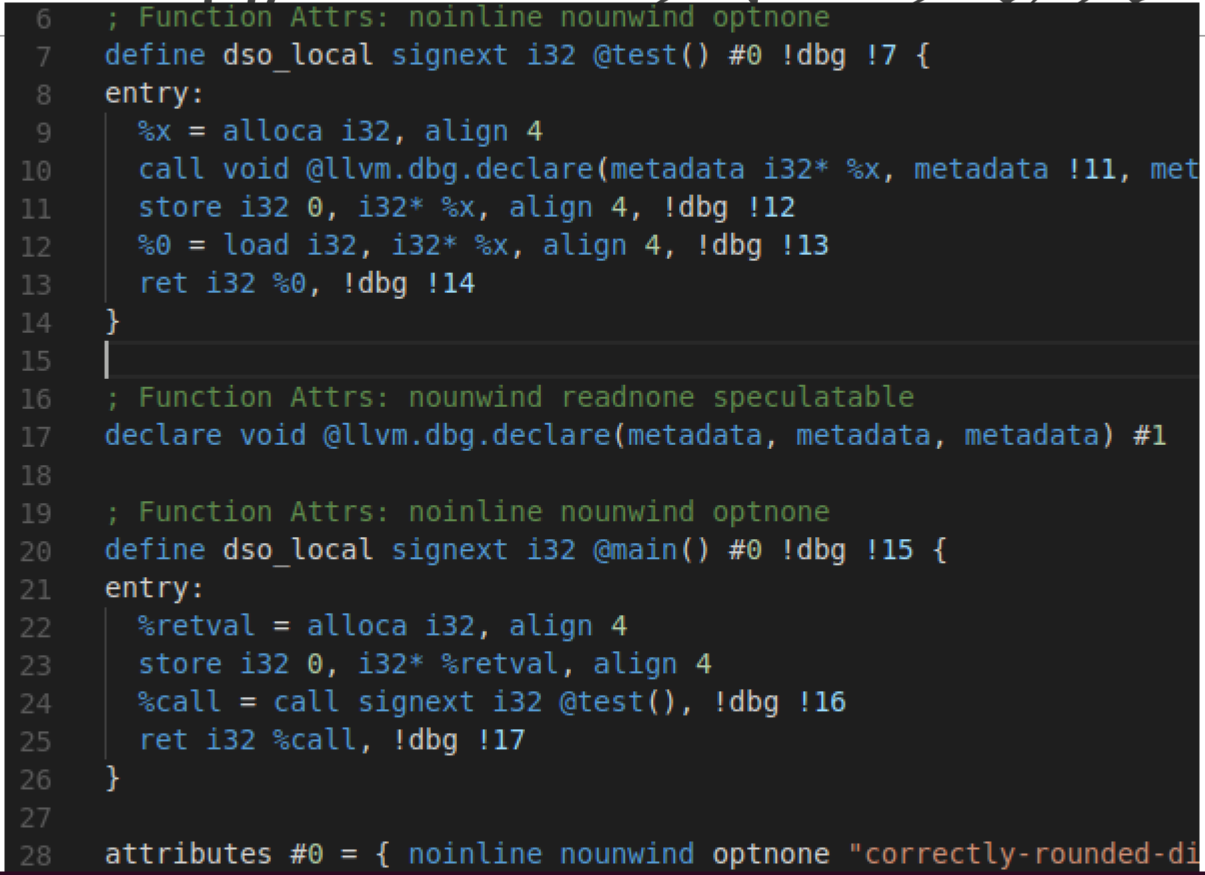
$$\begin{tikzcd}
\text{源码} \arrow[r] & {} \arrow[r] & \textbf{hv6.ll} \arrow[r] & \textbf{hv6.py} \arrow[d] \\
 & & \textbf{spec} \arrow[r] & \text{验证}
\end{tikzcd}$$

使用riscv相关clang/llvm工具：失败

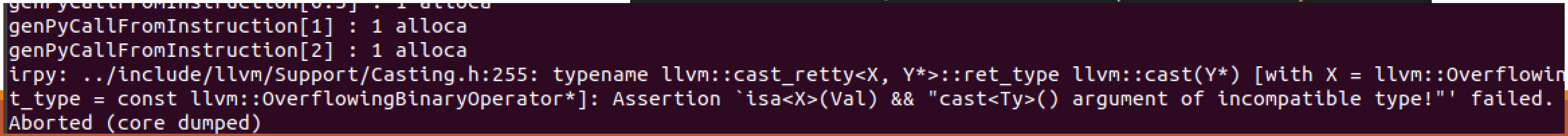


main.c　main.ll　test.sh

```c

int test()
{
    int x = 0;
    return x;
}

int main()
{
    return test();
}

```



```llvm
; Function Attrs: noinline nounwind optnone
define dso_local signext i32 @test() #0 !dbg !7 {
entry:
  %x = alloca i32, align 4
  call void @llvm.dbg.declare(metadata i32* %x, metadata !11, met
  store i32 0, i32* %x, align 4, !dbg !12
  %0 = load i32, i32* %x, align 4, !dbg !13
  ret i32 %0, !dbg !14
}

; Function Attrs: nounwind readnone speculatable
declare void @llvm.dbg.declare(metadata, metadata, metadata) #1

; Function Attrs: noinline nounwind optnone
define dso_local signext i32 @main() #0 !dbg !15 {
entry:
  %retval = alloca i32, align 4
  store i32 0, i32* %retval, align 4
  %call = call signext i32 @test(), !dbg !16
  ret i32 %call, !dbg !17
}

attributes #0 = { noinline nounwind optnone "correctly-rounded-di
```



```
genPyCallFromInstruction[0.5] : 1 alloca
genPyCallFromInstruction[1] : 1 alloca
genPyCallFromInstruction[2] : 1 alloca
irpy: ../include/llvm/Support/Casting.h:255: typename llvm::cast_retty<X, Y*>::ret_type llvm::cast(Y*) [with X = llvm::Overflowin
t_type = const llvm::OverflowingBinaryOperator*]: Assertion `isa<X>(Val) && "cast<Ty>() argument of incompatible type!"' failed.
Aborted (core dumped)
```

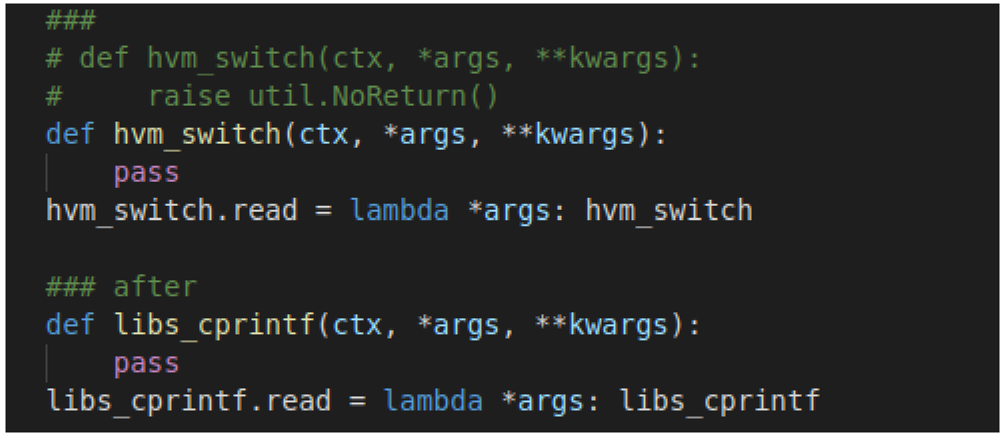
决定采用x86下的相关工具

14/50：直接跑

sys_map_pml4	512
sys_map_page_desc	512
sys_map_proc	512
sys_map_dev	512
sys_map_file	512
sys_alloc_pdpt	512
sys_alloc_pd	512
sys_alloc_pt	512
sys_alloc_frame	512
sys_copy_frame	0
sys_protect_frame	512
sys_free_pdpt	512
sys_free_pd	512
sys_free_pt	512
sys_free_frame	512
sys_reclaim_page	0
clone_proc	512
sys_set_proc_name	512

30/50：在spec中修改/添加函数定义

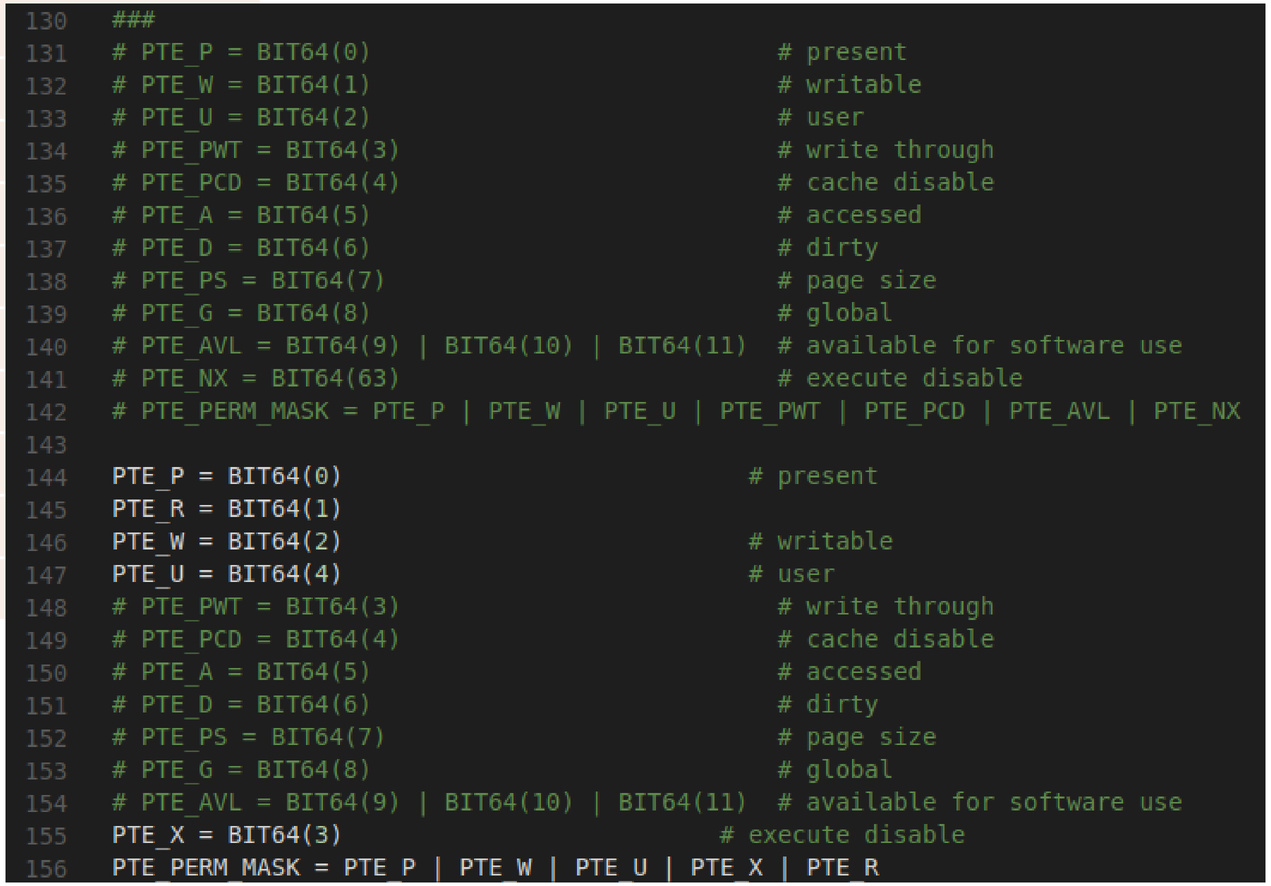
clone_proc	0	->	libs_cprintf
sys_set_proc_name	0	->	libs_cprintf
sys_set_runnable	0		
switch_proc	0	->	hvm_switch
sys_kill	0		
sys_reap	0		
sys_reparent	0		
send_proc	0		
recv_proc	0		
reply_wait_proc	0	->	libs_cprintf
call_proc	0	->	libs_cprintf
sys_create	0	->	libs_cprintf
sys_close	0	->	libs_cprintf
sys_dup	0	->	libs_cprintf
sys_dup2	0	->	libs_cprintf
sys_lseek	0	->	libs_cprintf
sys_map_pcipage	512		
sys_alloc_iommu_root	0	->	libs_cprintf



```python
###
# def hvm_switch(ctx, *args, **kwargs):
#     raise util.NoReturn()
def hvm_switch(ctx, *args, **kwargs):
    pass
hvm_switch.read = lambda *args: hvm_switch

### after
def libs_cprintf(ctx, *args, **kwargs):
    pass
libs_cprintf.read = lambda *args: libs_cprintf
```

38/50：在spec中修改页表标识位

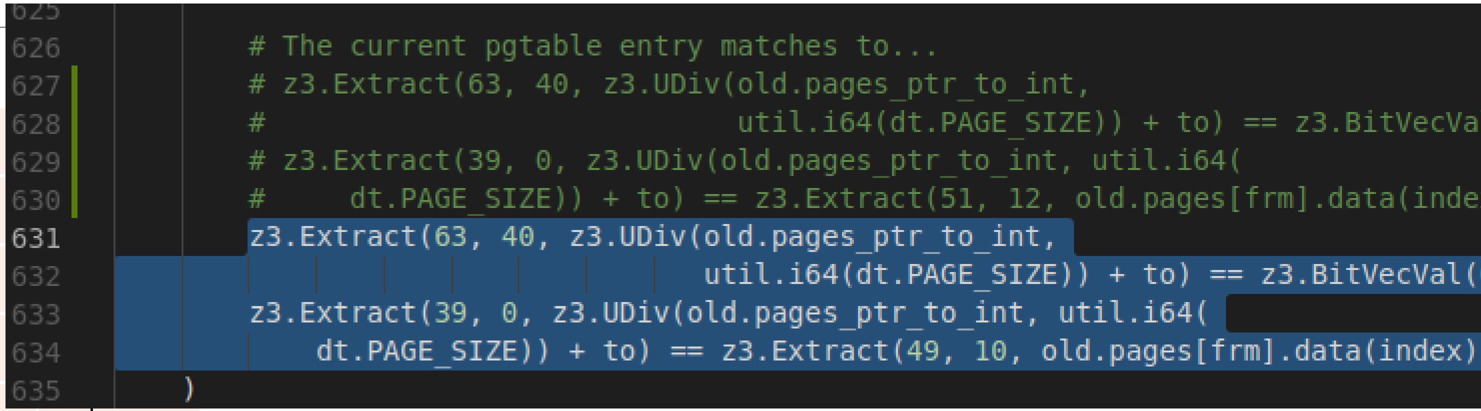
sys_map_pml4	0	->	PTE
sys_map_page_desc	0	->	PTE
sys_map_proc	0	->	PTE
sys_map_dev	0	->	PTE
sys_map_file	0	->	PTE
sys_alloc_pdpt	512		
sys_alloc_pd	512		
sys_alloc_pt	512		
sys_alloc_frame	0	->	PTE
sys_copy_frame	0		
sys_protect_frame	512		



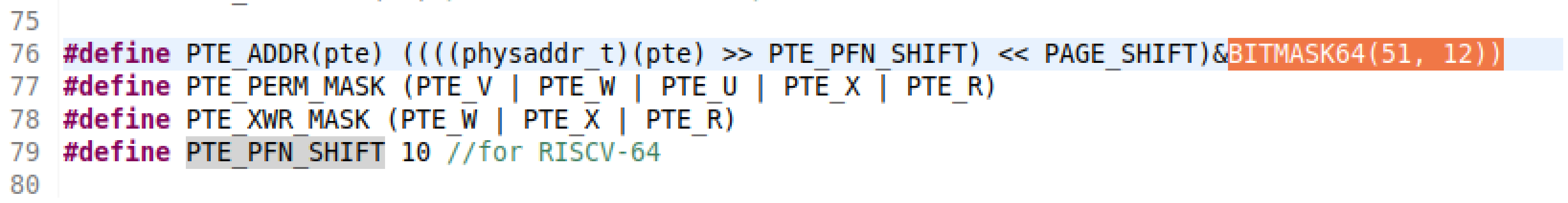
```
###
# PTE_P = BIT64(0)                            # present
# PTE_W = BIT64(1)                            # writable
# PTE_U = BIT64(2)                            # user
# PTE_PWT = BIT64(3)                          # write through
# PTE_PCD = BIT64(4)                          # cache disable
# PTE_A = BIT64(5)                            # accessed
# PTE_D = BIT64(6)                            # dirty
# PTE_PS = BIT64(7)                           # page size
# PTE_G = BIT64(8)                            # global
# PTE_AVL = BIT64(9) | BIT64(10) | BIT64(11)  # available for software use
# PTE_NX = BIT64(63)                          # execute disable
# PTE_PERM_MASK = PTE_P | PTE_W | PTE_U | PTE_PWT | PTE_PCD | PTE_AVL | PTE_NX

PTE_P = BIT64(0)                            # present
PTE_R = BIT64(1)
PTE_W = BIT64(2)                            # writable
PTE_U = BIT64(4)                            # user
# PTE_PWT = BIT64(3)                          # write through
# PTE_PCD = BIT64(4)                          # cache disable
# PTE_A = BIT64(5)                            # accessed
# PTE_D = BIT64(6)                            # dirty
# PTE_PS = BIT64(7)                           # page size
# PTE_G = BIT64(8)                            # global
# PTE_AVL = BIT64(9) | BIT64(10) | BIT64(11)  # available for software use
PTE_X = BIT64(3)                          # execute disable
PTE_PERM_MASK = PTE_P | PTE_W | PTE_U | PTE_X | PTE_R
```

43/50：修改spec的相关页表，修改hv6相关页表函数

sys_alloc_frame	0->	PTE	
sys_copy_frame	0		
sys_protect_frame	0->	spec&mmu	
sys_free_pdpt	0->	spec&mmu	
sys_free_pd	0->	spec&mmu	mask
sys_free_pt	0->	spec&mmu	mask
sys_free_frame	0->	spec&mmu	mask
sys_reclaim_page	0		



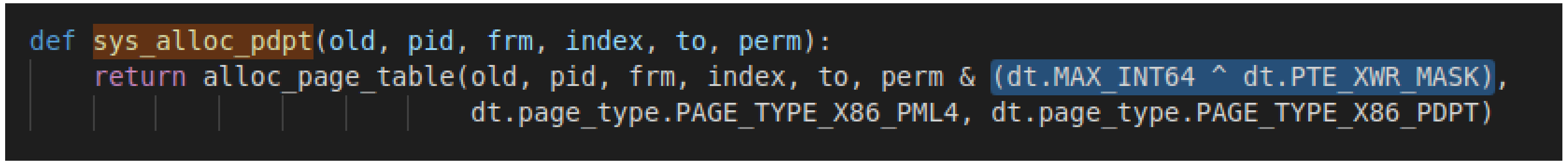
```
    # The current pgtable entry matches to...
    # z3.Extract(63, 40, z3.UDiv(old.pages_ptr_to_int,
    #                util.i64(dt.PAGE_SIZE)) + to) == z3.BitVecVa
    # z3.Extract(39, 0, z3.UDiv(old.pages_ptr_to_int, util.i64(
    #     dt.PAGE_SIZE)) + to) == z3.Extract(51, 12, old.pages[frm].data(inde
    z3.Extract(63, 40, z3.UDiv(old.pages_ptr_to_int,
                 util.i64(dt.PAGE_SIZE)) + to) == z3.BitVecVal(
    z3.Extract(39, 0, z3.UDiv(old.pages_ptr_to_int, util.i64(
        dt.PAGE_SIZE)) + to) == z3.Extract(49, 10, old.pages[frm].data(index)
)
```



```
#define PTE_ADDR(pte) ((((physaddr_t)(pte) >> PTE_PFN_SHIFT) << PAGE_SHIFT)&BITMASK64(51, 12))
#define PTE_PERM_MASK (PTE_V | PTE_W | PTE_U | PTE_X | PTE_R)
#define PTE_XWR_MASK (PTE_W | PTE_X | PTE_R)
#define PTE_PFN_SHIFT 10 //for RISCV-64
```

46/50：增加spec中对于XWR标志位的约束

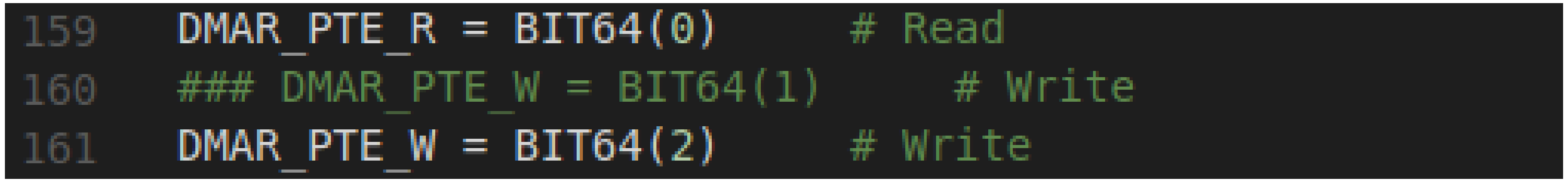
sys_map_file	0->	PTE
sys_alloc_pdpt	0->	PTE_XWR_MASK
sys_alloc_pd	0->	PTE_XWR_MASK
sys_alloc_pt	0->	PTE_XWR_MASK
sys_alloc_frame	0->	PTE



```
def sys_alloc_pdpt(old, pid, frm, index, to, perm):
    return alloc_page_table(old, pid, frm, index, to, perm & (dt.MAX_INT64 ^ dt.PTE_XWR_MASK),
                            dt.page_type.PAGE_TYPE_X86_PML4, dt.page_type.PAGE_TYPE_X86_PDPT)
```

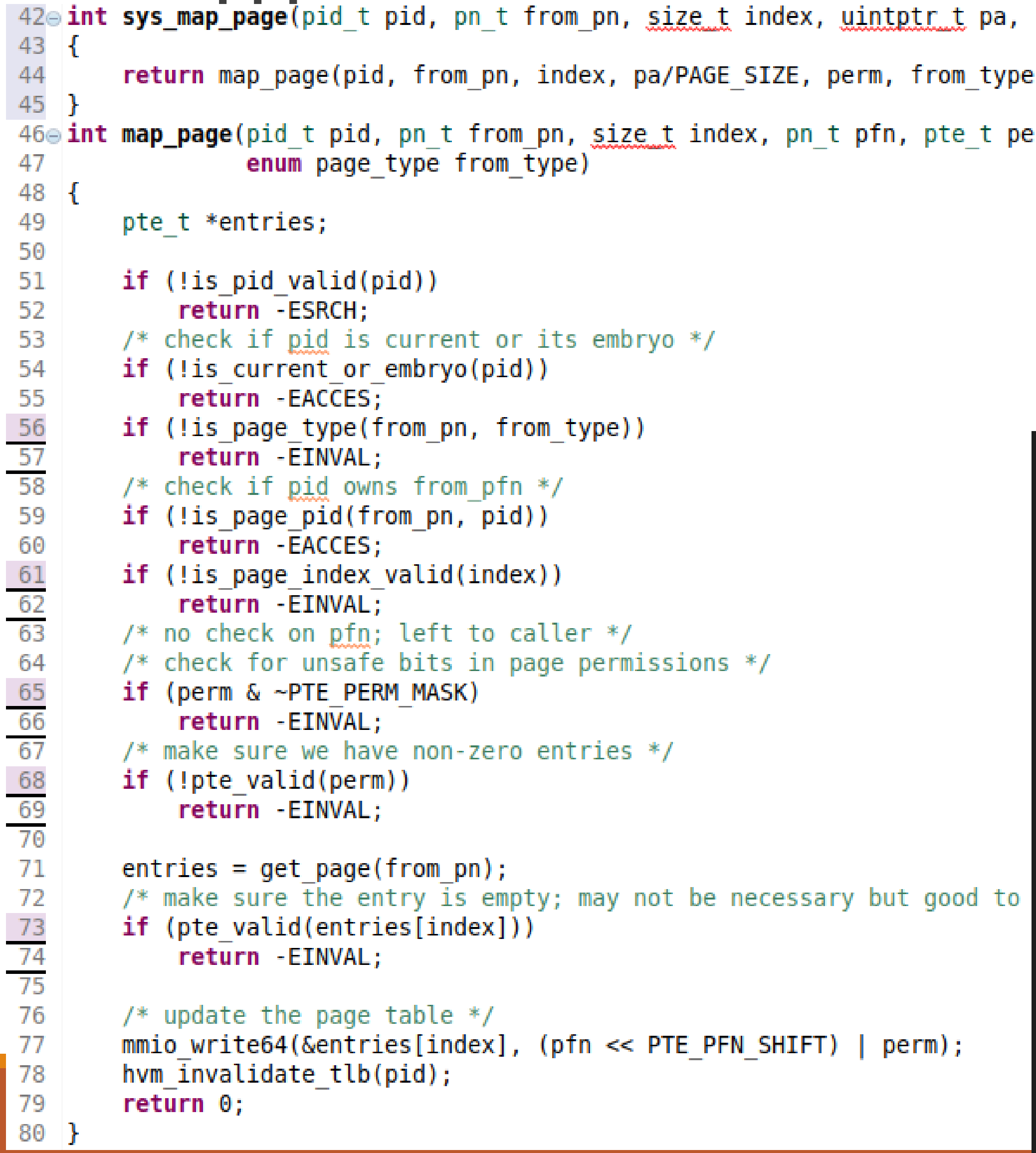
50/50：修改spec中iommu的页表标识位

sys_map_pcipage	0 ->	PTE
sys_alloc_iommu_root	0 ->	libs_cprintf
sys_alloc_iommu_pdpt	0 ->	DMAR_PTE_W
sys_alloc_iommu_pd	0 ->	DMAR_PTE_W
sys_alloc_iommu_pt	0 ->	DMAR_PTE_W
sys_alloc_iommu_frame	0 ->	DMAR_PTE_W
sys_map_iommu_frame	0 ->	PTE

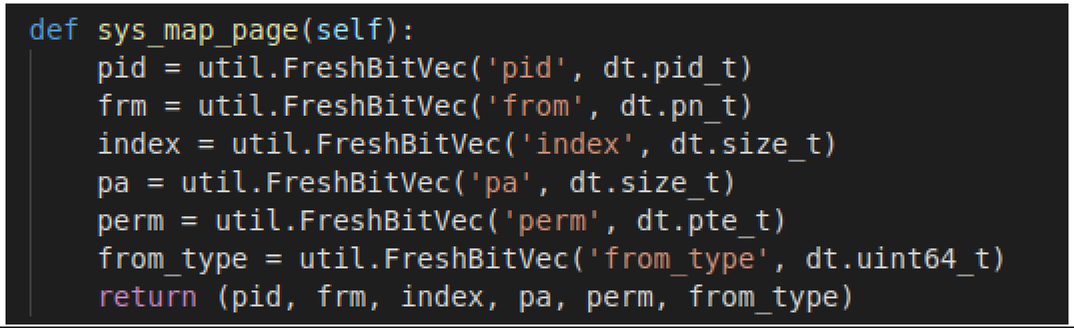


```
DMAR_PTE_R = BIT64(0)       # Read
### DMAR_PTE_W = BIT64(1)       # Write
DMAR_PTE_W = BIT64(2)       # Write
```

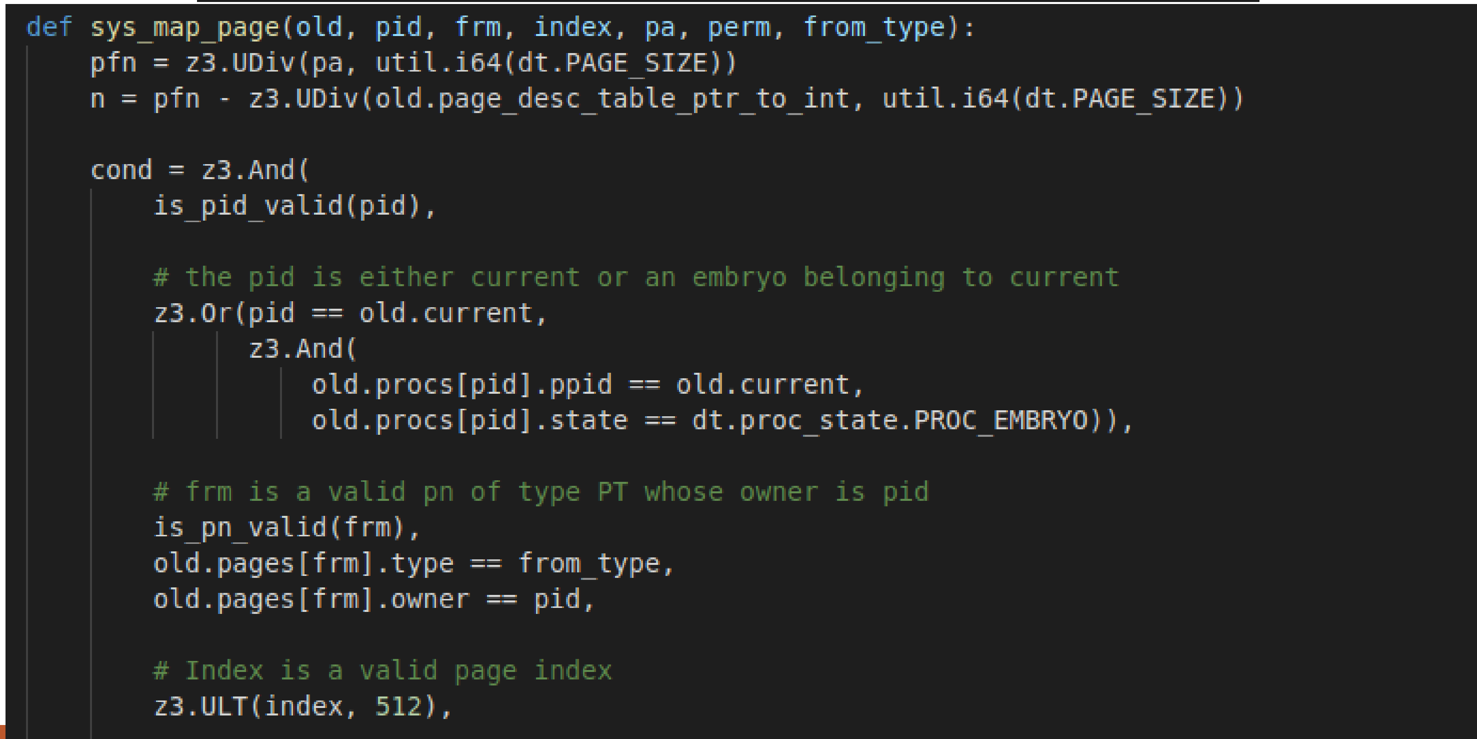
51/50：在spec中添加新系统调用的约束



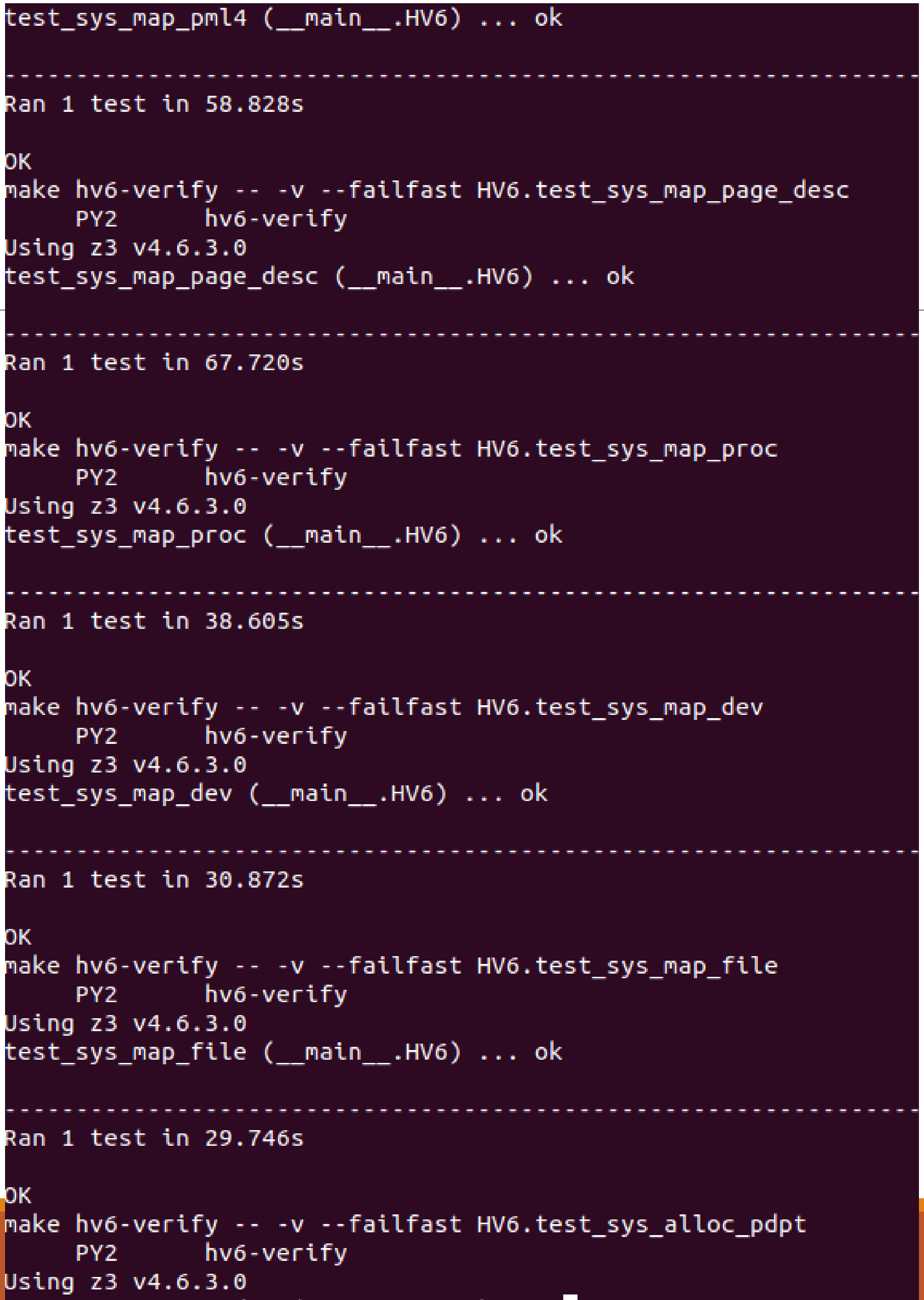
```c
int sys_map_page(pid_t pid, pn_t from_pn, size_t index, uintptr_t pa, pte_t perm, enum page_type from_type)
{
    return map_page(pid, from_pn, index, pa/PAGE_SIZE, perm, from_type);
}
int map_page(pid_t pid, pn_t from_pn, size_t index, pn_t pfn, pte_t perm,
             enum page_type from_type)
{
    pte_t *entries;

    if (!is_pid_valid(pid))
        return -ESRCH;
    /* check if pid is current or its embryo */
    if (!is_current_or_embryo(pid))
        return -EACCES;
    if (!is_page_type(from_pn, from_type))
        return -EINVAL;
    /* check if pid owns from_pfn */
    if (!is_page_pid(from_pn, pid))
        return -EACCES;
    if (!is_page_index_valid(index))
        return -EINVAL;
    /* no check on pfn; left to caller */
    /* check for unsafe bits in page permissions */
    if (perm & ~PTE_PERM_MASK)
        return -EINVAL;
    /* make sure we have non-zero entries */
    if (!pte_valid(perm))
        return -EINVAL;

    entries = get_page(from_pn);
    /* make sure the entry is empty; may not be necessary but good to
    if (pte_valid(entries[index]))
        return -EINVAL;

    /* update the page table */
    mmio_write64(&entries[index], (pfn << PTE_PFN_SHIFT) | perm);
    hvm_invalidate_tlb(pid);
    return 0;
}
```



```python
def sys_map_page(self):
    pid = util.FreshBitVec('pid', dt.pid_t)
    frm = util.FreshBitVec('from', dt.pn_t)
    index = util.FreshBitVec('index', dt.size_t)
    pa = util.FreshBitVec('pa', dt.size_t)
    perm = util.FreshBitVec('perm', dt.pte_t)
    from_type = util.FreshBitVec('from_type', dt.uint64_t)
    return (pid, frm, index, pa, perm, from_type)
```

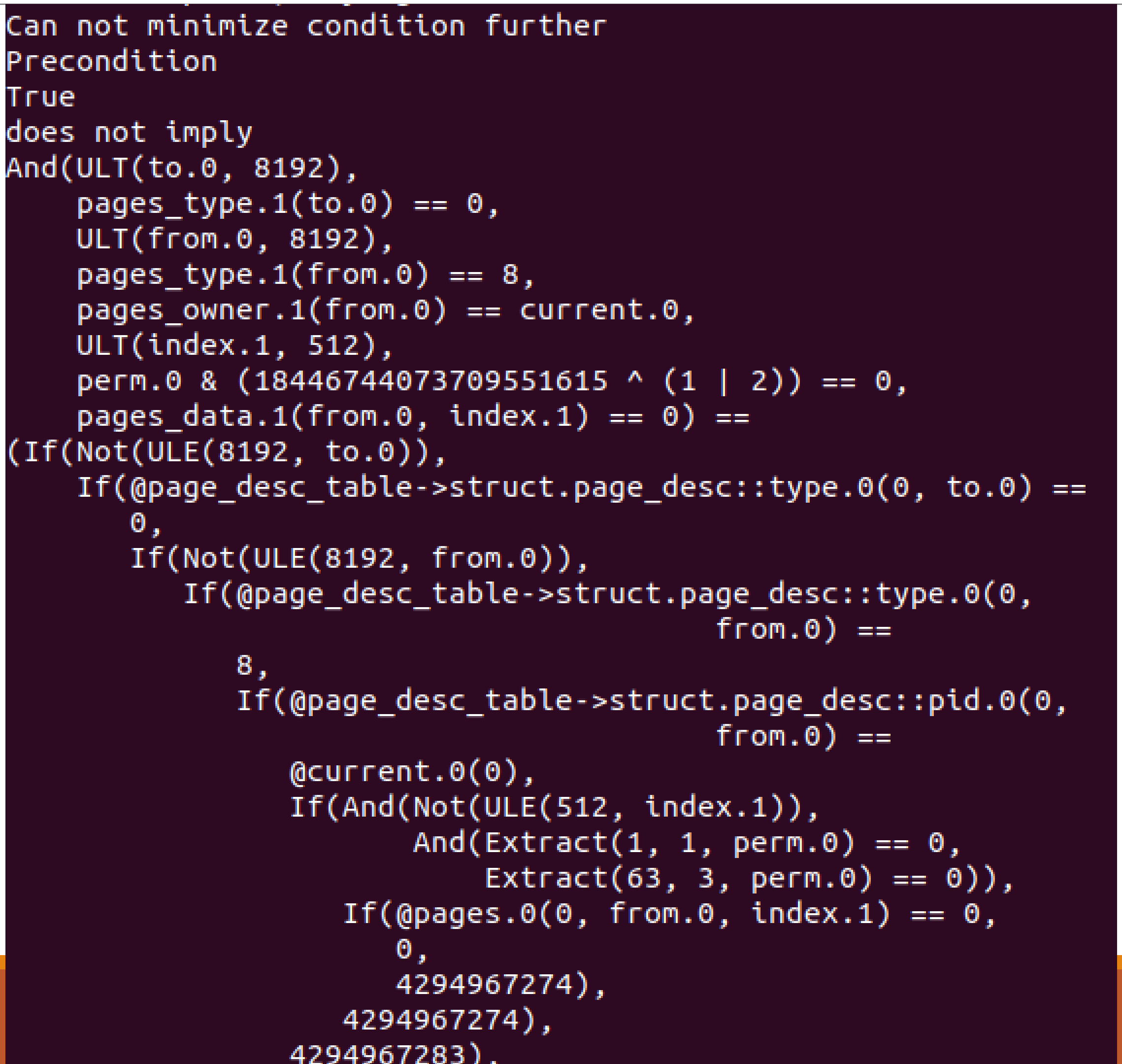


```python
def sys_map_page(old, pid, frm, index, pa, perm, from_type):
    pfn = z3.UDiv(pa, util.i64(dt.PAGE_SIZE))
    n = pfn - z3.UDiv(old.page_desc_table_ptr_to_int, util.i64(dt.PAGE_SIZE))

    cond = z3.And(
        is_pid_valid(pid),

        # the pid is either current or an embryo belonging to current
        z3.Or(pid == old.current,
              z3.And(
                  old.procs[pid].ppid == old.current,
                  old.procs[pid].state == dt.proc_state.PROC_EMBRYO)),

        # frm is a valid pn of type PT whose owner is pid
        is_pn_valid(frm),
        old.pages[frm].type == from_type,
        old.pages[frm].owner == pid,

        # Index is a valid page index
        z3.ULT(index, 512),
```



```
test_sys_map_pml4 (__main__.HV6) ... ok

----------------------------------------------------------------------
Ran 1 test in 58.828s

OK
make hv6-verify -- -v --failfast HV6.test_sys_map_page_desc
     PY2      hv6-verify
Using z3 v4.6.3.0
test_sys_map_page_desc (__main__.HV6) ... ok

----------------------------------------------------------------------
Ran 1 test in 67.720s

OK
make hv6-verify -- -v --failfast HV6.test_sys_map_proc
     PY2      hv6-verify
Using z3 v4.6.3.0
test_sys_map_proc (__main__.HV6) ... ok

----------------------------------------------------------------------
Ran 1 test in 38.605s

OK
make hv6-verify -- -v --failfast HV6.test_sys_map_dev
     PY2      hv6-verify
Using z3 v4.6.3.0
test_sys_map_dev (__main__.HV6) ... ok

----------------------------------------------------------------------
Ran 1 test in 30.872s

OK
make hv6-verify -- -v --failfast HV6.test_sys_map_file
     PY2      hv6-verify
Using z3 v4.6.3.0
test_sys_map_file (__main__.HV6) ... ok

----------------------------------------------------------------------
Ran 1 test in 29.746s

OK
make hv6-verify -- -v --failfast HV6.test_sys_alloc_pdpt
     PY2      hv6-verify
Using z3 v4.6.3.0
```

吐槽：错误信息极不友善



```
Can not minimize condition further
Precondition
True
does not imply
And(ULT(to.0, 8192),
    pages_type.1(to.0) == 0,
    ULT(from.0, 8192),
    pages_type.1(from.0) == 8,
    pages_owner.1(from.0) == current.0,
    ULT(index.1, 512),
    perm.0 & (18446744073709551615 ^ (1 | 2)) == 0,
    pages_data.1(from.0, index.1) == 0) ==
(If(Not(ULE(8192, to.0)),
    If(@page_desc_table->struct.page_desc::type.0(0, to.0) ==
       0,
       If(Not(ULE(8192, from.0)),
          If(@page_desc_table->struct.page_desc::type.0(0,
                                        from.0) ==
             8,
             If(@page_desc_table->struct.page_desc::pid.0(0,
                                        from.0) ==
                @current.0(0),
                If(And(Not(ULE(512, index.1)),
                       And(Extract(1, 1, perm.0) == 0,
                           Extract(63, 3, perm.0) == 0)),
                   If(@pages.0(0, from.0, index.1) == 0,
                      0,
                      4294967274),
                   4294967274),
                4294967283),
```

吐槽：错误信息极不友善

调试方式：二分删除spec和hv6的代码

但是不是任何时候都有效