

rv64ilp32: Run ILP32 on RV64 ISA

Guo Ren <guoren@kernel.org>

rv32ilp32 v.s. rv64lp64 v.s. rv64ilp32

	rv32ilp32	rv64lp64	rv64ilp32
ISA	rv32	rv64	rv64
long	32	64	32
pointer	32	64	32
char	8	8	8
short	16	16	16
int	32	32	32
long long	64	64	64

Patches Overview

```
[RFC PATCH V2 00/38] rv64ilp32: Running ILP32 on RV64 ISA
2023-11-12 6:15 UTC (39+ messages)
`-[RFC PATCH V2 01/38] riscv: u64ilp32: Unify vdso32 & compat_vdso into vdso/Makefile
`-[RFC PATCH V2 02/38] riscv: u64ilp32: Remove compat_vdso/
`-[RFC PATCH V2 03/38] riscv: u64ilp32: Add time-related vdso common flow for vdso32
`-[RFC PATCH V2 04/38] riscv: u64ilp32: Introduce ILP32 vdso for UXL=64
`-[RFC PATCH V2 05/38] riscv: u64ilp32: Adjust vdso kernel flow for 64ilp32 abi
`-[RFC PATCH V2 06/38] riscv: u64ilp32: Add signal support for compat
`-[RFC PATCH V2 07/38] riscv: u64ilp32: Add ptrace interface support
`-[RFC PATCH V2 08/38] riscv: u64ilp32: Adjust vdso alternative for 64ilp32 abi
`-[RFC PATCH V2 09/38] riscv: u64ilp32: Add xlen_t in user_regs_struct
`-[RFC PATCH V2 10/38] riscv: u64ilp32: Remove the restriction of UXL=32
`-[RFC PATCH V2 11/38] riscv: u64ilp32: Enable user space runtime switch
`-[RFC PATCH V2 12/38] riscv: s64ilp32: Unify ULL & UL into UXL in csr
`-[RFC PATCH V2 13/38] riscv: s64ilp32: Introduce xlen_t for 64ILP32 kernel
`-[RFC PATCH V2 14/38] riscv: s64ilp32: Add sbi support
`-[RFC PATCH V2 15/38] riscv: s64ilp32: Add asid support
`-[RFC PATCH V2 16/38] riscv: s64ilp32: Introduce PTR_L and PTR_S
`-[RFC PATCH V2 17/38] riscv: s64ilp32: Adjust TASK_SIZE for s64ilp32 kernel
`-[RFC PATCH V2 18/38] riscv: s64ilp32: Add ebpf jit support
`-[RFC PATCH V2 19/38] riscv: s64ilp32: Add ELF32 support
`-[RFC PATCH V2 20/38] riscv: s64ilp32: Add ARCH_RV64ILP32 Kconfig option
`-[RFC PATCH V2 21/38] riscv: s64ilp32: Add MMU_SV32 mode support
`-[RFC PATCH V2 22/38] riscv: s64ilp32: Add MMU_SV39 "
`-[RFC PATCH V2 23/38] riscv: s64ilp32: Enable native atomic64
`-[RFC PATCH V2 24/38] riscv: s64ilp32: Add TImode (128 int) support
`-[RFC PATCH V2 25/38] riscv: s64ilp32: Implement cmpxchg_double
`-[RFC PATCH V2 26/38] riscv: s64ilp32: Disable KVM
`-[RFC PATCH V2 27/38] riscv: s64ilp32: Correct the rv64ilp32 stackframe layout
`-[RFC PATCH V2 28/38] riscv: s64ilp32: Temporary workaround solution to gcc problem
`-[RFC PATCH V2 29/38] riscv: s64ilp32: Introduce ARCH_HAS_64ILP32_KERNEL for syscall
`-[RFC PATCH V2 30/38] riscv: s64ilp32: Add u32ilp32 ptrace support
`-[RFC PATCH V2 31/38] riscv: s64ilp32: Add u32ilp32 signal support
`-[RFC PATCH V2 32/38] riscv: s64ilp32: Validate harts by architecture name
`-[RFC PATCH V2 33/38] riscv: s64ilp32: Add rv64ilp32_defconfig
`-[RFC PATCH V2 34/38] riscv: Cleanup rv32_defconfig
`-[RFC PATCH V2 35/38] clocksource: riscv: s64ilp32: Use __riscv_xlen instead of CONFIG_32BIT
`-[RFC PATCH V2 36/38] irqchip: "
`-[RFC PATCH V2 37/38] add tinylab defconfig
`-[RFC PATCH V2 38/38] 64ilp32 v.s. 64lp64
```

PATCH [01 - 11] u64ilp32

PATCH [12 - 36] s64ilp32

PATCH [37 - 38] ilp32 v.s. lp64

<https://lore.kernel.org/linux-riscv/20231112061514.2306187-1-quoren@kernel.org/>

s64ilp32 + u32ilp32:
Fedora rv32 ready, about 1800+
packages.

s64ilp32 + u64ilp32:
Only console & SPEC_CPU test.

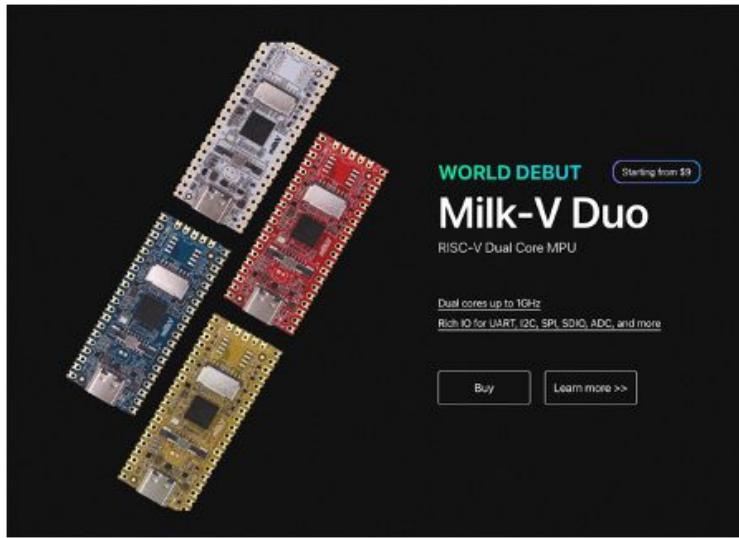
Agenda

- Motivation Discussion
- Implementation Discussion

Motivation

64MB rv64 ISA Chips

There have been x86-x32, mips-n32, arm64ilp32 existed, why introduce rv64ilp32?



128Mb Ox64 SBC

Community price: \$8.00

CORE	<ul style="list-style-type: none">• 64-bit 480MHz RV64 C906• 32-bit 320MHz RV32 E907• 32-bit 150MHz RV32 E902
MEMORY	<ul style="list-style-type: none">• 728KB internal SRAM• 64MB internal PSRAM• 128Mb (16MB) XSPI Flash• microSD slot

Memory Footprint

rv64lp64 has 25% more memory footprint than rv64ilp32 (16MB Linux)!

Calculation Process:

$$\text{rv64ilp32} = (4096 - 3406) = 690$$

$$\text{rv64lp64} = (4096 - 3231) = 865$$

$$(865 - 690)/690 = 25\%$$

sizeof(xxx)	ILP32	LP64
struct page	32	64
list_head	8	16
hlist_node	8	16
vm_area_struct	68	136

```
Platform Name : riscv-virtio_qemu
Platform Features : medleg
Platform HART COUNT : 1
Platform IPI Device : aclkint-mswi
Platform Timer Device : aclkint-timer@ 100000000Hz
Platform Memory Controller : uort#8250
Platform HDA Device : ...
Platform PMU Device : ...
Platform reboot Device : sifive_test
Platform Shutdown Device : ...
Platform Suspend Device : ...
Platform CPU Device : ...
Firmware Base : 0x10000000
Firmware Size : 368 KB
Firmware RW Offset : 0x40000
Runtime SBI Version : 1.0
```

```
  * Name          : riscv-virtio_gemu
  * Features     : 
  * Model        : 
  * Version      : 0.0.1
  * Uart Port    : 
  * IPI Device   : dclint=mask
  * Memory Model : direct=100000000Hz
  * Console Device: 
  * RMU Device   : 
  * SMMU Device  : 
  * Robot Device : 
  * Sifive Test  : sifive_test
  * Suspend Device: 
  * Secure Device: 
  * Base Address : 0x60000000
  * Size         : 368 KB
  * Offset       : 0x00000000
  * SBT Version  : 1.8.7
```

```
WT ID : 0
WT Model : p05
WT Priv Version : v1.32
WT Base ISA : x86_64mtfdch
WT ISA Extensions : time,ssic
WT Model ID : 0
WT PMP Granularity : 4
WT PMP Address Bits: 54
WT PMP Granularity : 16
WT MDELEG : 0x0000000000000166
WT MDELEG : 0x0000000000000509
```

```
Ignoring memory virtio-ram@0x80000000 - 0x804000000  
model: riscv-virtualmem  
Implementation ID=0x1 Version@<0x1002  
extension detected  
extension@0x0  
extension@0x1  
IMC extension detected  
IMC extension detected  
reserved mem: 0xaaaaaaaaaaaaaaaa...0xaaaaaaaaaaaaaaaaffff (256 KIB) msp non-reusable mmode_res1@0x80000000
```

```
o-init: stack:off, heap alloc/off, heap free/off  
1377K/1684K available  
o-234X kernel mem: 129K rwdtio, 129K rodtio, 161K init, 207K bss, 2608K reserved, 0K cmc-reserved  
o-Malign4, Order=0, MinNbPages=0, CPUs=1, Nodes=1  
o-64, nr_irqs: 64, preallocated_irqs: 0
```

```
olice<00000000> mapped 95 interrupts with 1 handlers for 2 contexts.  
source: riscv_clocksource: mask: 0xfffffffffffff max_cycles: 0x8e601710, max_idle_ns: 440795202120 ns  
clock: 64 bits at 10Mhz, resolution 100ns, wraps every 439804651100hns  
time: Timer interrupt in S-mode is available via vsys extension
```

```
hash table entries: 512 (order: 0, 4096 bytes, linear)
coche hash table entries: 512 (order: 0, 4096 bytes, linear)
locator using 16 bits (65536 entries)
```

```
[ 0.000000] source: jiffies; mask: 0xffffffff max_clocksource: 0xffffffff, max_idle_ns: 7645041785100000 ns
[ 0.000000] source: Switched to clocksource riscv_clocksource
[ 0.000000] set: timestamp_bits=62 max_order=12 bucket_order=0
[ 0.000000] 8250/16550 driver, 4 ports, IRQ sharing disabled
```

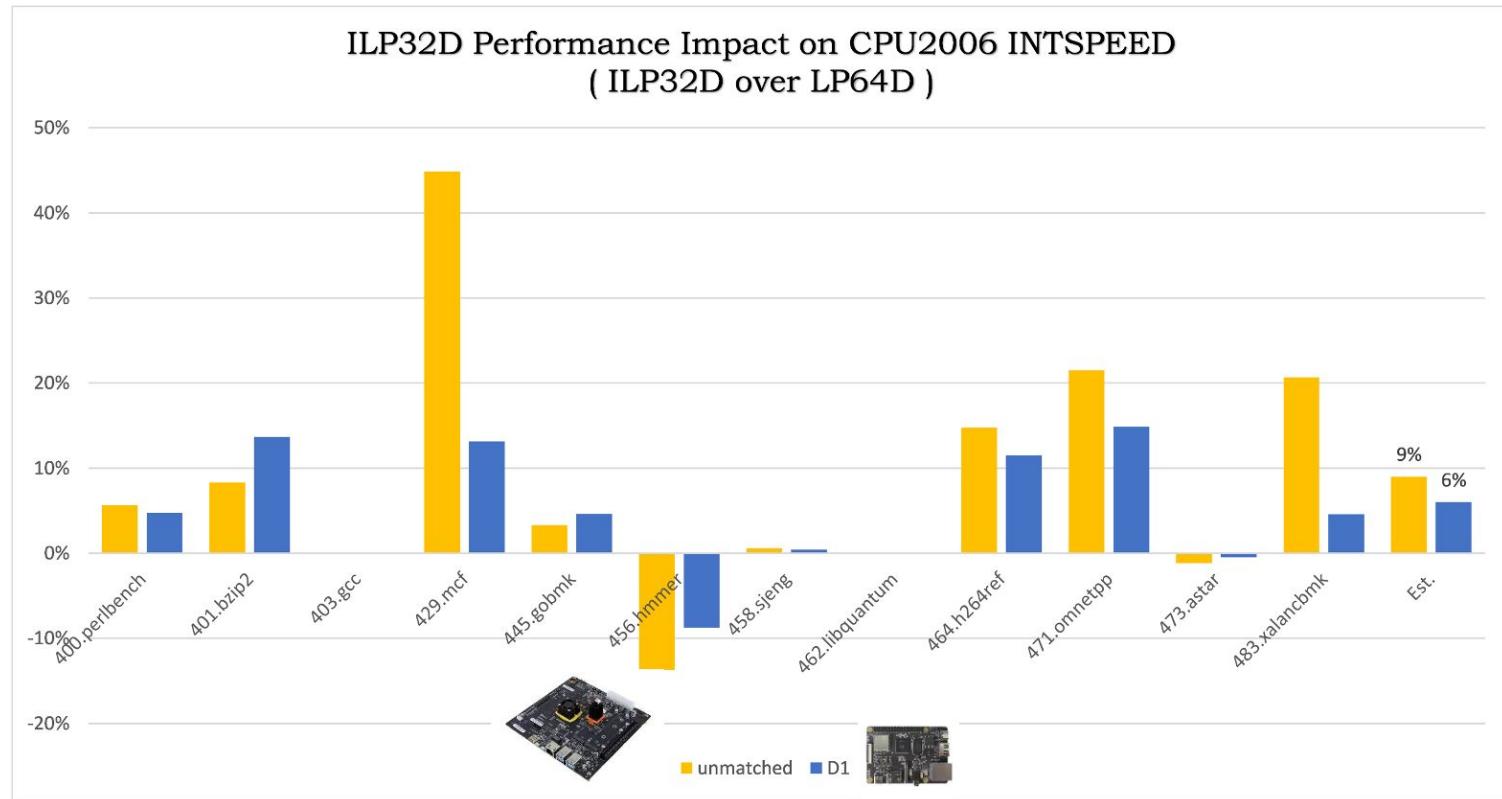
```
[0].serial: ttyS0 at MMIO 0x10000000 (irq = 2, base_baud = 230400) is a 16550A
  console: (ttyS0) enabled
  scaling unused clocks
  using rootwatch; rootfs is invalid.
```

```
[    ] [unwind kernel image <initmem>] memory: 16KB  
[    ] [memory protection not selected by kernel config.  
[    ] [iminit as init process  
[    ] [cinit as init process
```

```
    .anon:0 inactive_anon:0 isolated_anon:0
    .file:0 inactive_file:0 isolated_file:0
```

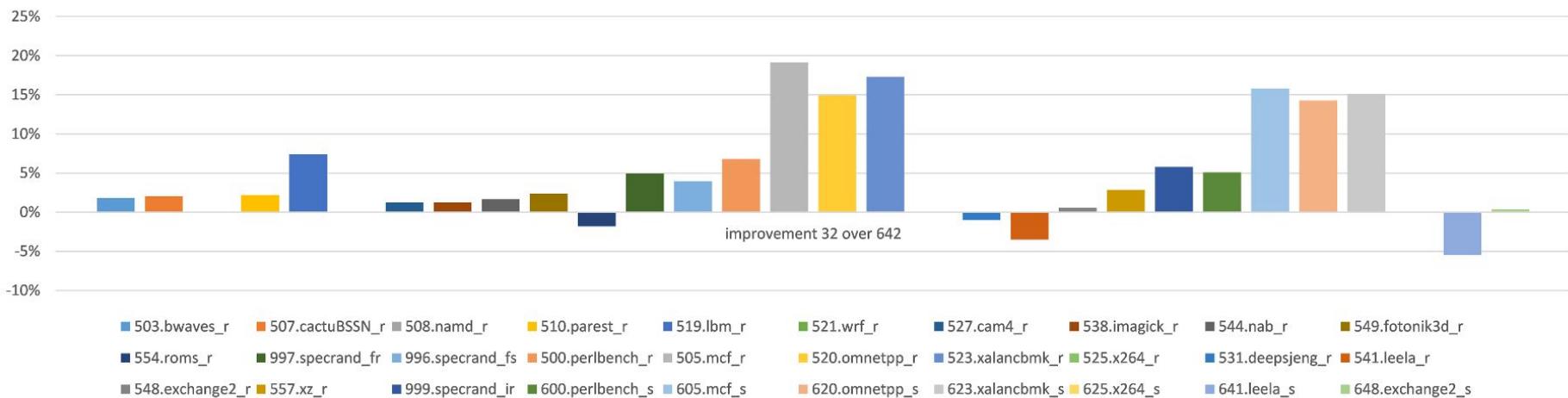
```
[23] file_pspp_35 free:cmd  
[24] file_pspp_35 active_anon:0kB active_file:0kB inactive_file:0kB unevictable:0kB isolated(anon):0kB isolated(file):0kB mapped:0kB dirty:0kB writeback:0kB mapped_file:0kB unevictable_file:0kB dirty_file:0kB writeback_file:0kB  
all_unreclaimable:0kB free:  
free:129248K boost:  
[25] file_min:468KB low:548KB high:700KB reserved_highatomic:0kB active_anon:0kB inactive_anon:0kB active_file:0kB inactive_file:0kB  
file_pspp_35 local_file:0kB free_file:0kB cmd:  
cmd:220K local_file:0kB free_file:0kB cmd:
```

SPEC CPU 2006



SPEC CPU 2017

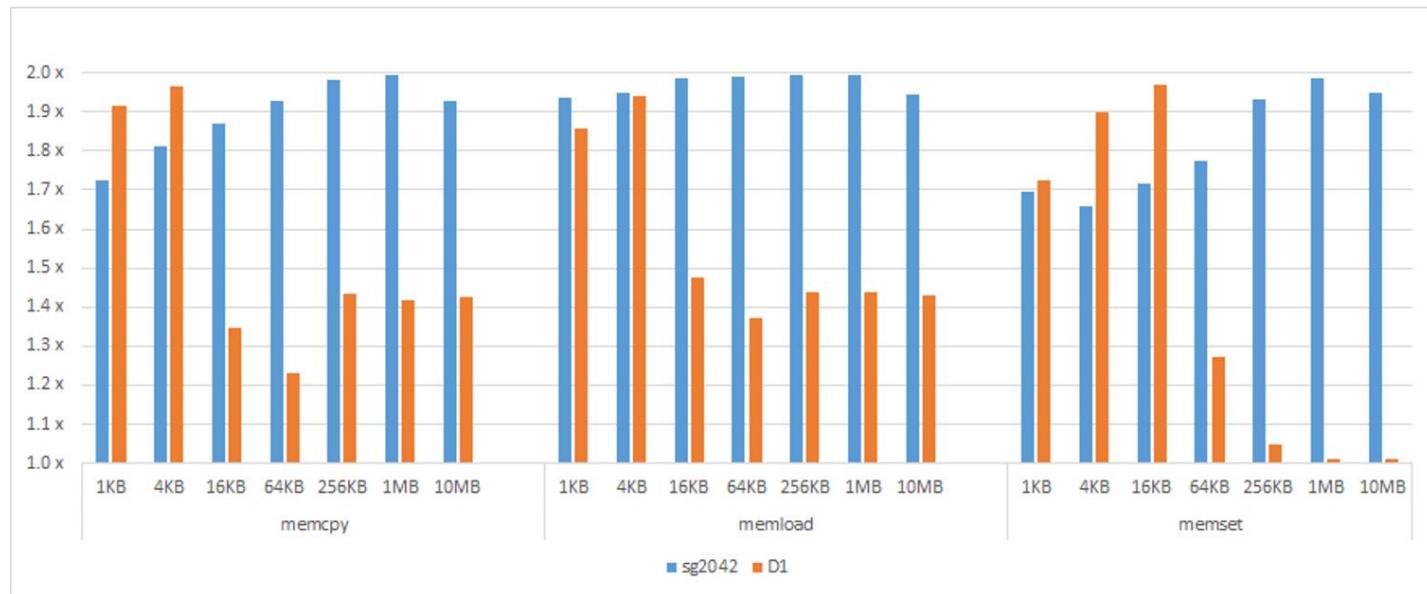
ILP32D Performance Impact on CPU2017
(ILP32D over LP64D)



arm32 -> rv64

Putting aside political factors, what value does rv64ilp32 bring to customers?

- memcpy/memload/memset performance in Linux kernel
- eBPF JIT
- Atomic64



Implementation

Stack layout Optimization

64bit -> 32bit

Callee saved the register width

For 64-bit ISA (including 64lp64, 64ilp32), callee can't determine the correct width used in the register, so they saved the maximum width of the ISA register, i.e., xlen size. We also found this rule in x86-x32, mips-n32, and aarch64ilp32, which comes from 64lp64. See PATCH [20]

Here are two downsides of this:

- It would cause a difference with 32ilp32's stack frame, and s64ilp32 reuses 32ilp32 software stack. Thus, many additional compatible problems would happen during the porting of 64ilp32 software.
- It also increases the budget of the stack usage.

```
<setup_vm>:  
    auipc    a3,0xff3fb  
    add     a3,a3,1234 # c0000000  
    li      a5,-1  
    lui     a4,0xc0000  
    addw   sp,sp,-96  
    srl    a5,a5,0x20  
    subw   a4,a4,a3  
    auipc   a2,0x111a  
    add    a2,a2,1212 # cldlf000  
    sd     s0,80(sp)----+  
    sd     s1,72(sp)  
    sd     s2,64(sp)  
    sd     s7,24(sp)  
    sd     s8,16(sp)  
    sd     s9,8(sp)      -> All <= 32b widths, but occupy 64b  
    sd     ra,88(sp)      stack space.  
    sd     s3,56(sp)      Affect memory footprint & cache  
    sd     s4,48(sp)      performance.  
    sd     s5,40(sp)  
    sd     s6,32(sp)  
    sd     s10,0(sp)----+  
    sll    a1,a4,0x20  
    subw   a2,a2,a3  
    and    a4,a4,a5
```

So here is a proposal to riscv 64ilp32 ABI:

- Let the compiler prevent callee saving ">32b variables" in callee-registers. (Q: We need to measure, how the influence of 64b variables cross function call?)

GCC problem

64-bit Optimization problem

There is an existing problem in 64ilp32 gcc that combines two pointers in one register. Liao is solving that problem. Before he finishes the job, we could prevent it with a simple `noinline` attribute, fortunately.

```
struct path {
    struct vfsmount *mnt;
    struct dentry *dentry;
} __randomize_layout;

struct nameidata {
    struct path      path;
    ...
    struct path      root;
}
} __randomize_layout;

struct nameidata *nd
...
nd->path = nd->root;
6c88          ld      a0,24(s1)
                ^// a0 contains two pointers
e088          sd      a0,0(s1)
                mntget(path->mnt);
                // Need "lw a0,0(s1)" or "a0 << 32; a0 >> 32"
2a6150ef      jal     c01ce946 <mntget> // bug!
```

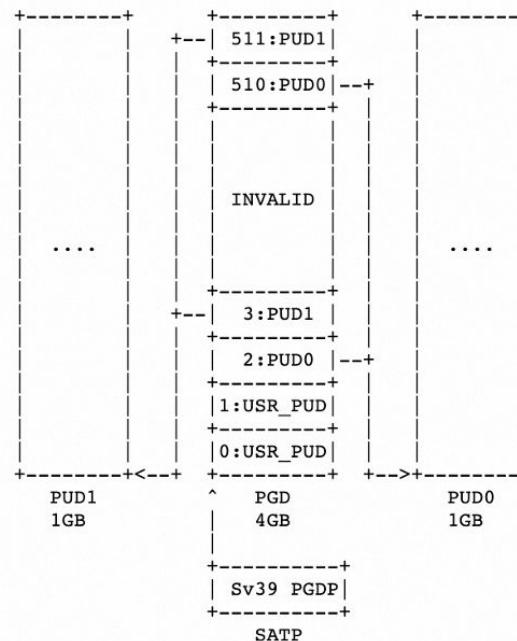
Sign-extend addressing

Old: zero-extend

New: sign-extend

The 64ilp32 gcc still uses sign-extend lw & auipc to generate address variables because inserting zero-extend instructions to mask the highest 32-bit would cause significant code size and performance problems. Thus, we invented an OS approach to solve the problem:

- When satp=bare and start physical address < 2GB, there is no sign-extend address problem.
- When satp=bare and start physical address > 2GB, we need zjmp like hardware extensions to mask high 32bit.
(Fortunately, all existed SoCs' (D1/D1s/F133, CV1800B, k230, BL808) start physical address < 2GB.)
- When satp=sv39, we invent double mapping to make the sign-extended virtual address the same as the zero-extended virtual address.



Thank you