平头哥 RISC-V

2023 RISC-V 中国峰会

打破 64ILP32 的魔咒
挣脱 32 位的桎梏

Guo Ren
guoren@kernel.org

近期工作:

- 在算能128核平台上实现 qspinlock
- 64ILP32

什么是 64ILP32?

(老)32位: 32ILP32　在 32位架构硬件上，运行 ILP32 数据类型软件
64位: 64LP64　在 64位架构硬件上，运行 LP64 数据类型软件
新32位: 64ILP32　在 64位架构硬件上，运行 ILP32 数据类型软件

- 为什么叫新32位?
- 如何衡量 32位 和 64位?

剧本：新32位 取代 老32位，与 64位 分天下 (Linux)

新32位的魔咒 (Linux)

在 Linux 过去的十几年中，各大架构都曾尝试新32位，x86 的 X32，MIPS 的 N32，和 ARM 的 64ilp32，但都没有成功。

在去年RISC-V峰会上，我分享了 COMPAT 特性的贡献，其中就有关于新32位ABI的讨论：
Linux 架构子系统的维护者 (Arnd Bergeman) 公开表示，不支持新32位ABI，且不会将 ARM64 的 新32位用户态功能合入内核主线

RISC-V psABI 副主席 Jessica 直言不讳：其他架构都不行，凭啥RV行？
Every single attempt at ILP32 ABI for a 64-bit architecture has failed to take off in the past, so I struggle to see why RV64 will be any different.
以上二位大佬的言论，代表的是主流观点：新32位行不通

但是，新32位有一种魔力，它吸引着一代又一代架构不断尝试，却又铩羽而归，这就是我们要打破的魔咒，一个经过十几年形成的观念。

We had the same debate for Arm64, and ended up never merging the ILP32 patches despite them being well written and maintainable, to limit the number of supported user space ABIs as well as the possible attack vectors when there is an exploitable bug that is specific to an ABI.

```
00/13] riscv: compat: Add COMPAT mode support for rv64
21 02:29:02 +0000   [thread overview]
Message-ID: <AA7091EA-C3AF-47CE-B0C5-E2ECE4133D09@jrtc27.com> (raw)
In-Reply-To: <CAJF2gTQ04utyOc9=q9Tu92togGuyygKgq3tWwfPBCuyTfLh2SQ@mail.gmail.com>

On 27 Dec 2021, at 01:16, Guo Ren <guoren@kernel.org> wrote:
>
> On Mon, Dec 27, 2021 at 4:31 AM Arnd Bergmann <arnd@arndb.de> wrote:
>>
>> On Sun, Dec 26, 2021 at 7:38 AM Guo Ren <guoren@kernel.org> wrote:
>>> On Sun, Dec 26, 2021 at 4:36 PM Jisheng Zhang <jszhang3@mail.ustc.edu.cn> wrote:
>>>> On Wed, 22 Dec 2021 20:59:30 +0800 Guo Ren <guoren@kernel.org> wrote:
>>>>> On Wed, Dec 22, 2021 at 2:10 AM Arnd Bergmann <arnd@arndb.de> wrote:
>>>>>
>>>>> What about adding RV64 ILP32 support instead? This don't need HW side
>>>>> modifications so can benefit all RV64.
>>>
>>> ILP32 is another topic in C Language Data Type Models and it couldn't
>>> replace the standard rv32 ecosystem.
>>> COMPAT is a common framework in Linux (7 arches have been supported),
>>> so let rv64 support COMPAT mode is considerable.
>>>
>>> Customers would choose ILP32 / RV32-compat by themself and that
>>> depends on which one has a better ecosystem.
>>
>> From a kernel perspective, supporting both is not much more work than
>> supporting either of them. We had the same debate for Arm64, and ended
>> up never merging the ILP32 patches despite them being well written
>> and maintainable, to limit the number of supported user space ABIs
>> as well as the possible attack vectors when there is an exploitable
>> bug that is specific to an ABI.
>>
>> arm64 does support big-endian mode, which is a similar niche, but it
>> can't easily be removed after it's already supported. Supporting normal
>> compat mode is the easiest here because it doesn't add another user
>> space ABI, but I'd strongly recommend not to add any other ones.
>
> @Palmer Dabbelt How do you think about supporting ILP32 & COMPAT both
> in rv64? And let users vote by foot which is better.

As psABI TG co-chair I really do not want an ILP32 RV64 to exist if it
can at all be avoided. Every single attempt at an ILP32 ABI for a
64-bit architecture has failed to take off in the past, so I struggle
to see why RV64 will be any different. So, in my opinion, there is a
relatively high barrier to entry for it to be an official frozen ABI,
and without it being that I doubt upstreams will want to go near it, be
it Linux, GCC, binutils or GCC, but they can speak for themselves if
they feel otherwise.

Also, with every year that goes by, ILP32 becomes more and more limited
in what you can use it for, due to increased memory footprints...

Jess
```

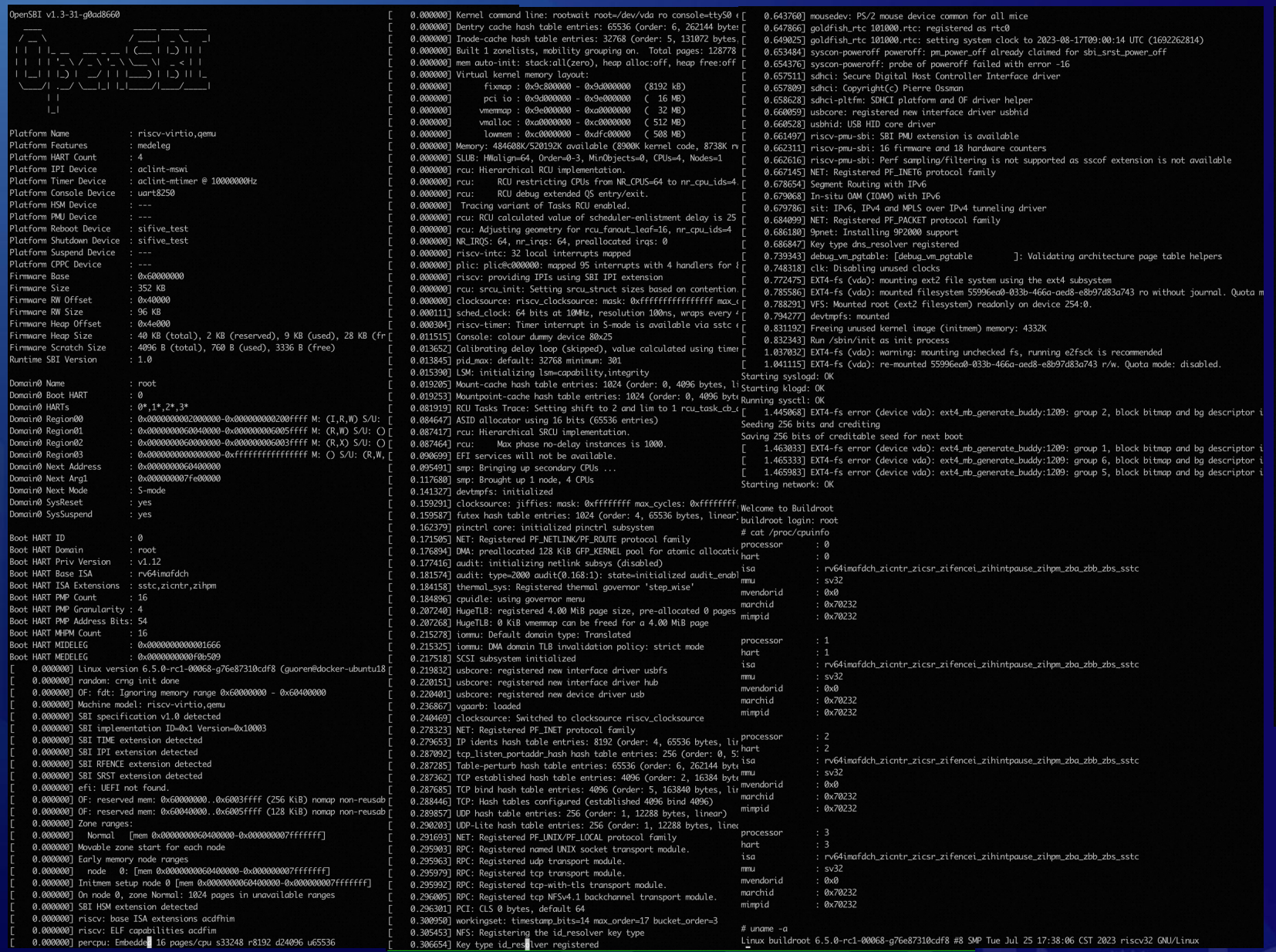
ARM64用户态支持新32位的内核补丁

ARM64 的 ILP32 历时2年 (2017-2019)，6 个人的努力，共 22 个补丁，高悬于 Catalin Marinas (ARM64 维护人) 的仓库中，不被合入。如果不汲取前辈的经验教训，不改良，不创新，我们还会重蹈覆辙!

Age	Commit message (Expand)	Author	Files	Lines
2019-06-27	arm64: ilp32: Make the Kconfig option default y staging/ilp32-5.1	Catalin Marinas	1	-0/+1
2019-06-27	arm64:ilp32: add ARM64_ILP32 to Kconfig	Andrew Pinski	1	-1/+8
2019-06-27	arm64:ilp32: add vdso-ilp32 and use for signal return	Philipp Tomsich	11	-14/+280
2019-06-27	arm64: ptrace: handle ptrace_request differently for aarch32 and ilp32	Yury Norov	1	-2/+19
2019-06-27	arm64: ilp32: introduce ilp32-specific sigframe and ucontext	Yury Norov	5	-1/+101
2019-06-27	arm64: signal32: move ilp32 and aarch32 common code to separated file	Yury Norov	5	-25/+54
2019-06-27	arm64: signal: share lp64 signal structures and routines to ilp32	Yury Norov	2	-279/+353
2019-06-27	arm64: ilp32: introduce syscall table for ILP32	Yury Norov	6	-4/+125
2019-06-27	arm64: ilp32: share aarch32 syscall handlers	Yury Norov	3	-102/+107
2019-06-27	arm64: ilp32: introduce binfmt_ilp32.c	Yury Norov	2	-0/+88
2019-06-27	arm64: change compat_elf_hwcap and compat_elf_hwcap2 prefix to a32	Yury Norov	5	-16/+16
2019-06-27	arm64: introduce binfmt_elf32.c	Yury Norov	5	-27/+39
2019-06-27	arm64: introduce AUDIT_ARCH_AARCH64ILP32 for ilp32	Yury Norov	2	-0/+4
2019-06-27	arm64: ilp32: add is_ilp32_compat_{task,thread} and TIF_32BIT_AARCH64	Yury Norov	3	-2/+32
2019-06-27	arm64: introduce is_a32_compat_{task,thread} for AArch32 compat	Yury Norov	15	-48/+84
2019-06-27	arm64: uapi: set __BITS_PER_LONG correctly for ILP32 and LP64	Andrew Pinski	1	-1/+8
2019-06-27	arm64: rename functions that reference compat term	Yury Norov	15	-93/+94
2019-06-27	arm64: rename COMPAT to AARCH32_EL0	Andrew Pinski	19	-38/+47
2019-06-27	arm64: ilp32: add documentation on the ILP32 ABI for ARM64	Yury Norov	1	-0/+52
2019-06-27	thread: move thread bits accessors to separated file	Yury Norov	3	-74/+89
2019-06-27	ptrace: Add compat PTRACE_{G,S}ETSIGMASK handlers	James Morse	1	-14/+38
2019-06-27	arm64: signal: Make parse_user_sigframe() independent of rt_sigframe layout	Dave Martin	1	-6/+8

相比前辈，我们做了哪些改良和创新?

首次让 新32位内核 运行在 64位指令架构 之上




```
irqchip: riscv: s64ilp32: Use __riscv_xlen instead of CONFIG_32BIT
clocksource: riscv: s64ilp32: Use __riscv_xlen instead of CONFIG_32BIT
riscv: Cleanup rv32_defconfig
riscv: s64ilp32: Add rv64ilp32_defconfig
riscv: s64ilp32: Validate harts by architecture name
riscv: s64ilp32: Add u32ilp32 signal support
riscv: s64ilp32: Add u32ilp32 ptrace support
riscv: s64ilp32: Introduce ARCH_HAS_64ILP32_KERNEL for syscall
riscv: s64ilp32: Temporary workaround solution to gcc problem
riscv: s64ilp32: Correct the rv64ilp32 stackframe layout
riscv: s64ilp32: Disable KVM
riscv: s64ilp32: Implement cmpxchg_double
riscv: s64ilp32: Add TImode (128 int) support
riscv: s64ilp32: Enable native atomic64
riscv: s64ilp32: Add MMU_SV39 mode support
riscv: s64ilp32: Add MMU_SV32 mode support
riscv: s64ilp32: Add ARCH_RV64ILP32 Kconfig option
riscv: s64ilp32: Add ELF32 support
riscv: s64ilp32: Add ebpf jit support
riscv: s64ilp32: Adjust TASK_SIZE for s64ilp32 kernel
riscv: s64ilp32: Introduce PTR_L and PTR_S
riscv: s64ilp32: Add asid support
riscv: s64ilp32: Add sbi support
riscv: s64ilp32: Introduce xlen_t for 64ILP32 kernel
riscv: s64ilp32: Unify ULL & UL into UXL in csr
riscv: u64ilp32: Enable user space runtime switch
riscv: u64ilp32: Remove the restriction of UXL=32
riscv: u64ilp32: Add xlen_t in user_regs_struct
riscv: u64ilp32: Adjust vDSO alternative for 64ilp32 abi
riscv: u64ilp32: Add ptrace interface support
riscv: u64ilp32: Add signal support for compat
riscv: u64ilp32: Adjust vDSO kernel flow for 64ilp32 abi
riscv: u64ilp32: Introduce ILP32 vdso for UXL=64
riscv: u64ilp32: Add time-related vDSO common flow for vdso32
riscv: u64ilp32: Remove compat_vdso/
riscv: u64ilp32: Unify vdso32_& compat_vdso into vdso/Makefile
```

内核

用户

支持完整新32位:

- u64ilp32 - 新32位 用户态
- s64ilp32 - 新32位 内核态

前辈只实现了新32位的用户态

栈的优化

- 前辈的方案，沿用 64位栈布局
- 我们的方案，改用 32位栈布局

效果：

- 栈的内存开销减少一半
- 入栈性能，提升 3%
  (减半无效数据，减轻了 CPU 写队列的负担)
- 出栈性能，提升 1%
  (减半无效数据，减轻缓存预取的负担)

简易测量: 直接对比出入栈的 IPC

TODO:
砍掉一半数据，受限于测试方法，观测到的性能提升很小。
只有找到 cache miss 的零界点，才能凸显性能优势。

```
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 # c1d1f000
     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?)
```

符号扩展寻址方案

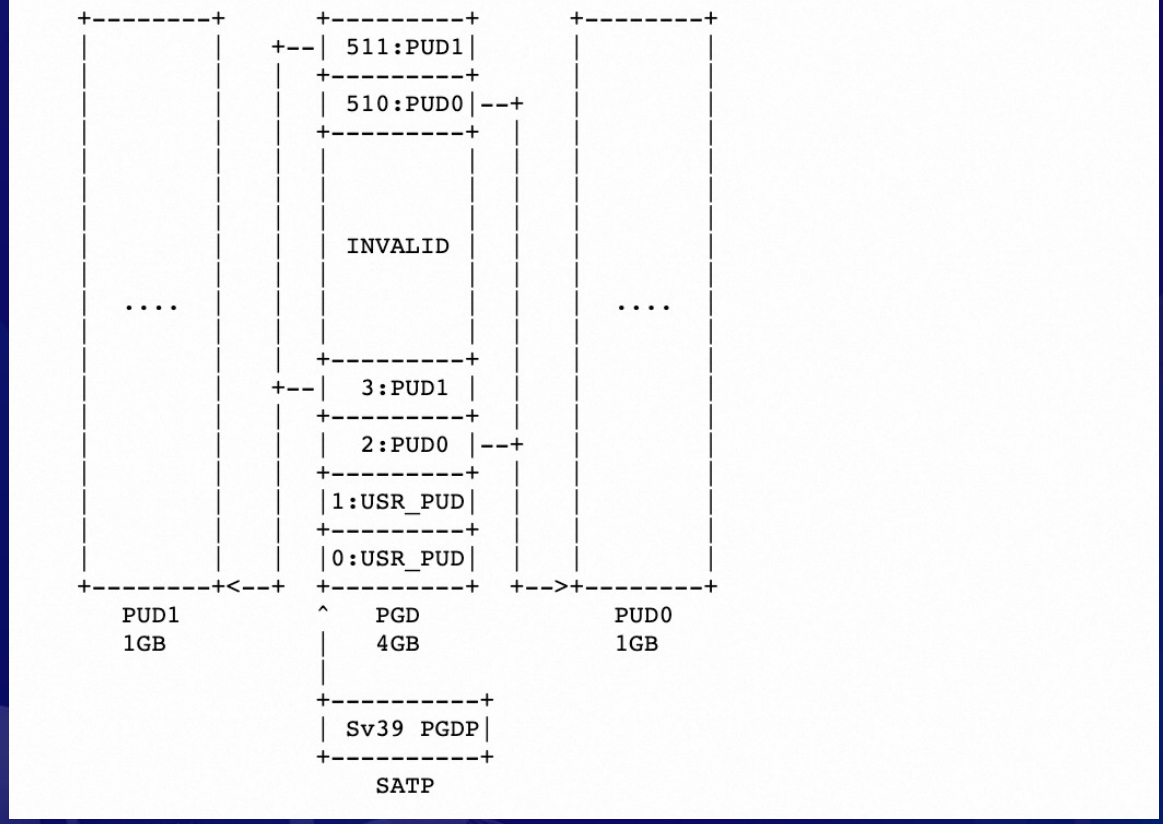
- 前辈方案：零扩展寻址
  缺点：需要插入额外指令对地址高位抹零，影响性能和代码密度。

- 我们方案：符号扩展寻址
  用 OS 实现 MMU 的双重映射，使得符号扩展的地址，和零扩展的地址，在取指和访存效果上相同。

- 优势：
  - 减轻编译器工作
  - 提升性能
  - 提升代码密度

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 zjpm liked 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.



新32位是一个大工程

GCC: 廖仕华，陈嘉炜
性能分析: 王俊强，陈小欧
FEDORA: 张松松，傅炜
QEMU: 李威威
LLVM: 陆旭凡，廖春玉
LINUX: 李春强，郭任

后勤保障: ISCAS 吴伟，T-HEAD 熊健

第一层价值: 比较数据类型 (ILP32:LP64)

新32位 v.s. 64位

使用小数据类型 (INT8/FP8) 让 AI 推理获得了不菲的性能和成本收益，那么 ILP32 相比 LP64 能否达到一样的效果?

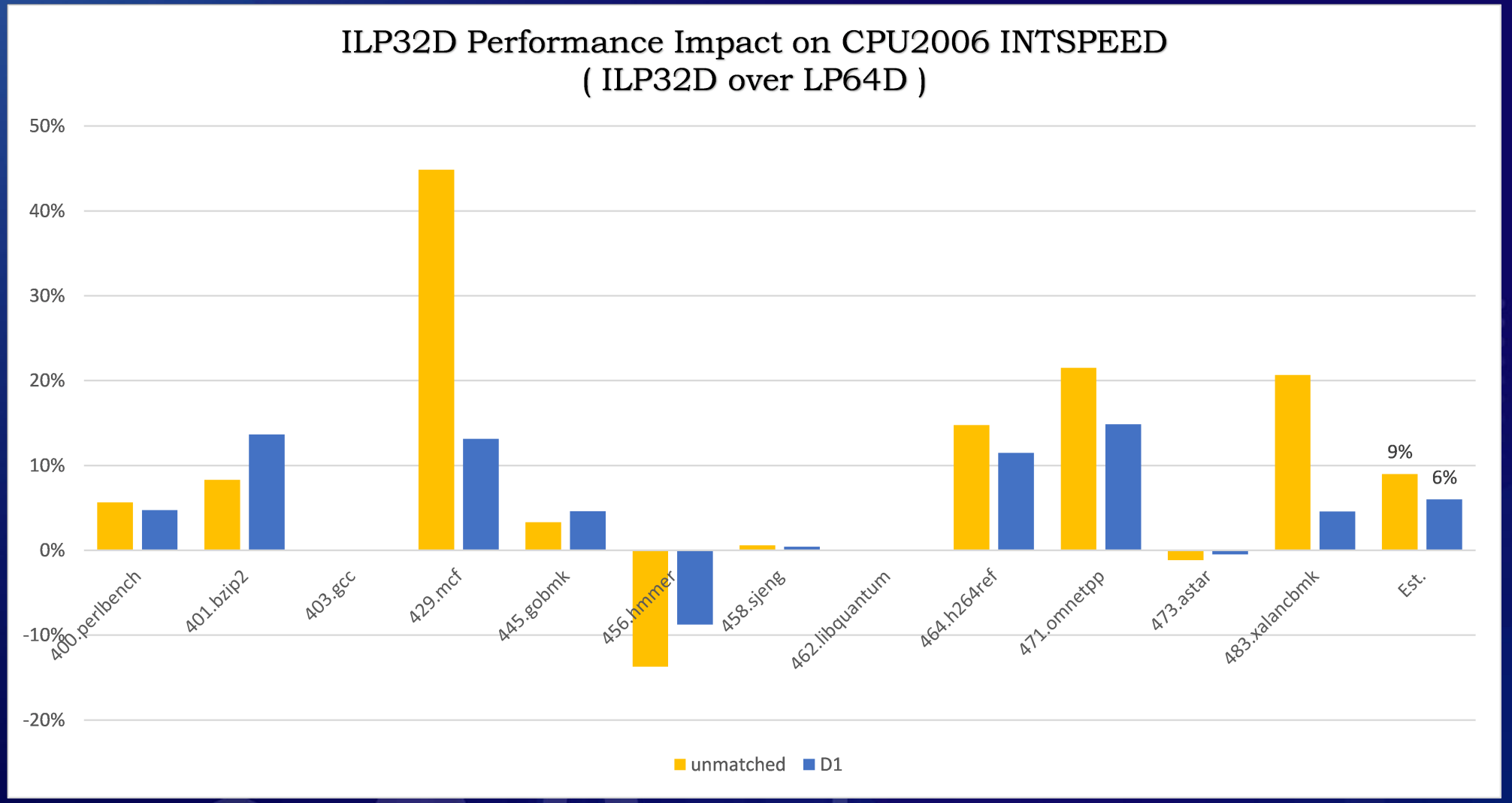
SPEC CPU2006 最大提升 40% (32 位时代)

测试平台:

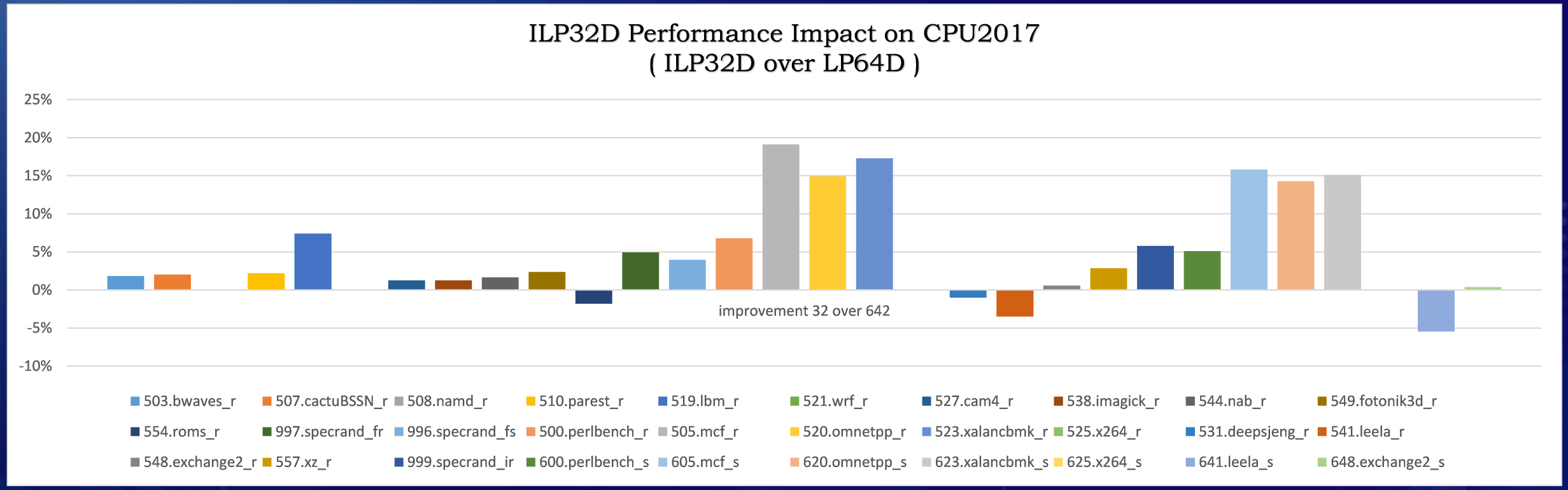
- UNMATCHED

- D1

TODO:
解决编译器插入无用符号扩展指令，导致指令数增加的问题



SPEC CPU2017 最大提升 18% (64 位时代)



因为 D1 内存限制，所以只测试 UNMATCHED 平台，
感谢 SPEC 小队的工作。

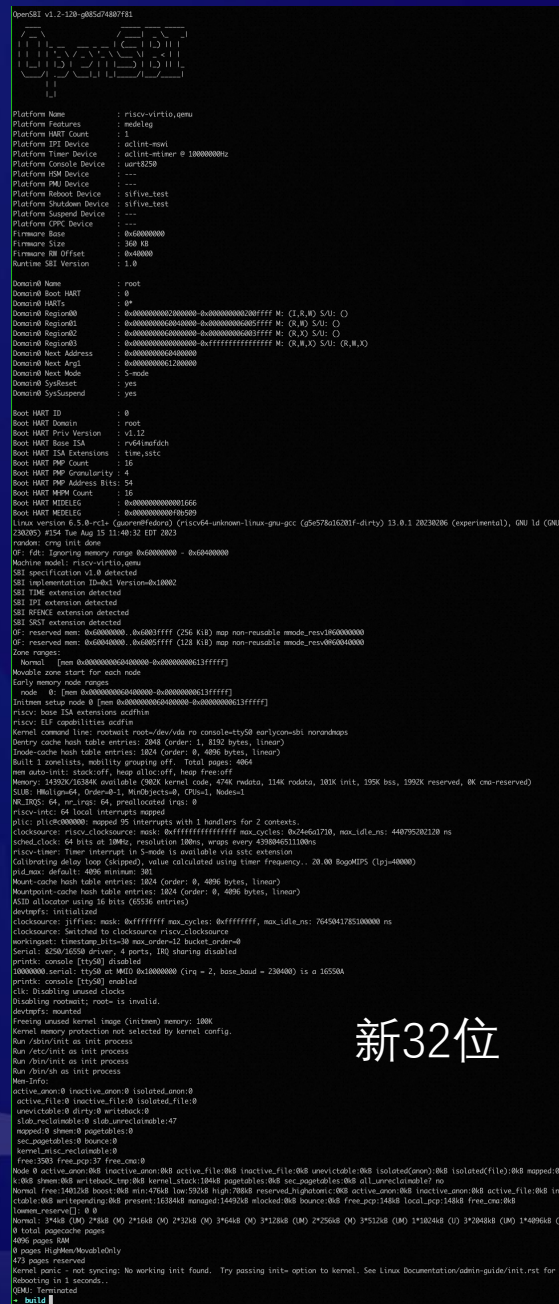
64位 内存开销比 新32位 多 28%

基于 tinyconfig 16MB 内存，并尽量消除 64位和32位的 kconfig 差异， 启动日志逐行对齐。

计算公式：
内存开销 = TotalPages - FreePages
64位　　= 4096 - 3334 = 762
新32位　= 4096 - 3503 = 593
(762-593)/593 = 169/593 = 28.49%

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

感谢泰晓社区提供的内核配置。更多内容，请参考泰晓社区为 RISC-V 架构研发的小型化系统项目：
https://tinylab.org/tinylinux (8MB 内存跑 Linux)



新32位

64位

新32位 普惠所有 RISC-V 64位硬件平台

目前市面上在售的 RISC-V Linux 硬件都是纯 64位架构，这是新32位落地 RISC-V 的天时！

第二层价值: 比较指令架构 (RV64:RV32)

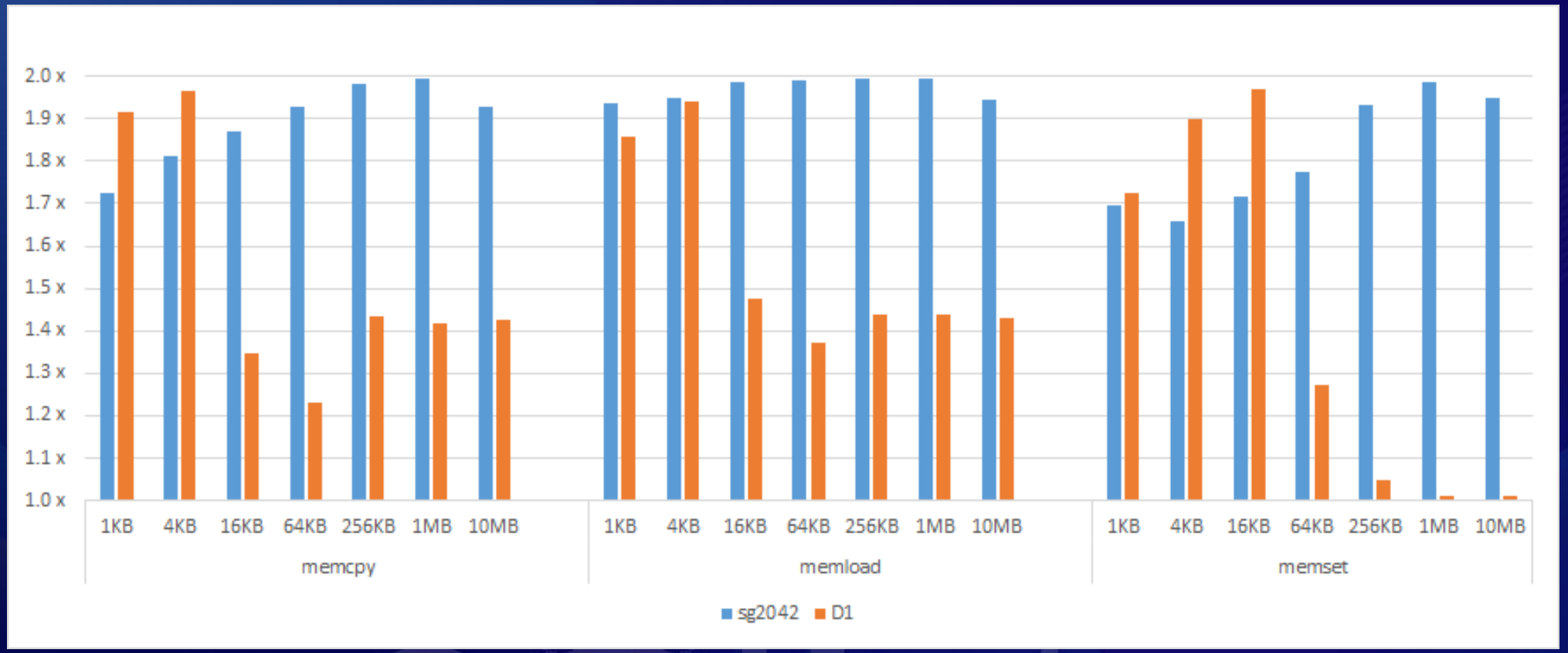
新32位 v.s. 老32位

因为用户态的扩展指令集稀释了 64位相对 32位的架构优势，
所以我们实现了新32位内核，下面在内核态比较它们。

第一拳：新32位内核的访存优势

Linux/arch/riscv/lib/memcpy.S

基于 RV64 ISA 的新32位内核拥有双字访存指令 LD/SD，而基于 RV32 ISA 的老32位内核仅有单字访存指令 LW/SW，导致老32位内核访存性能大幅落后新32位。内核访存能力直接决定网络性能，系统性能，和驱动性能。。。



第二拳：老32位缺失内核功能

- eBPF JIT：
  eBPF 是当下的热门技术，其生态正在快速生长，它可以不改内核源码或加载内核模块的前提下，扩展内核功能。JIT 是即时编译加速技术，将 eBPF 码编译成原生指令再执行，而 eBPF 寄存器是64位宽，无法直接映射32位指令架构，很多64位算术操作也不能简单映射到32位架构的指令上。所以，只有 64位指令架构才能使 eBPF JIT 真正达到原生内核的性能。

- ATOMIC64：
  原生 64 位原子指令，能够和 32位原子操作互斥。但 32位指令架构没有64位原子指令，只能通过软件模拟（GENERIC_ATOMIC64），无法和32位原子指令互斥，存在语义错误。

第三拳: 新32位拓展内核能力

DCAS（双元比较交换）指令，是非常昂贵的硬件实现。在 32 位 Linux 中，只有 x86 实现了这一功能，它可以开启 SLUB 加速的特性：

mm/slub.c：

```
if (s->flags & __CMPXCHG_DOUBLE) {
        ret = __update_freelist_fast(slab, freelist_old, counters_old,
                                     freelist_new, counters_new);
    } else {
        ret = __update_freelist_slow(slab, freelist_old, counters_old,
                                     freelist_new, counters_new);
    }
```

新32位 Linux 内核通过双字 LR/SC 指令实现 DCAS，达成了相同效果。

应用处理器架构的发展趋势

- ARMv9 只有 64位指令架构

- RVA Profiles 也只有 64位指令架构

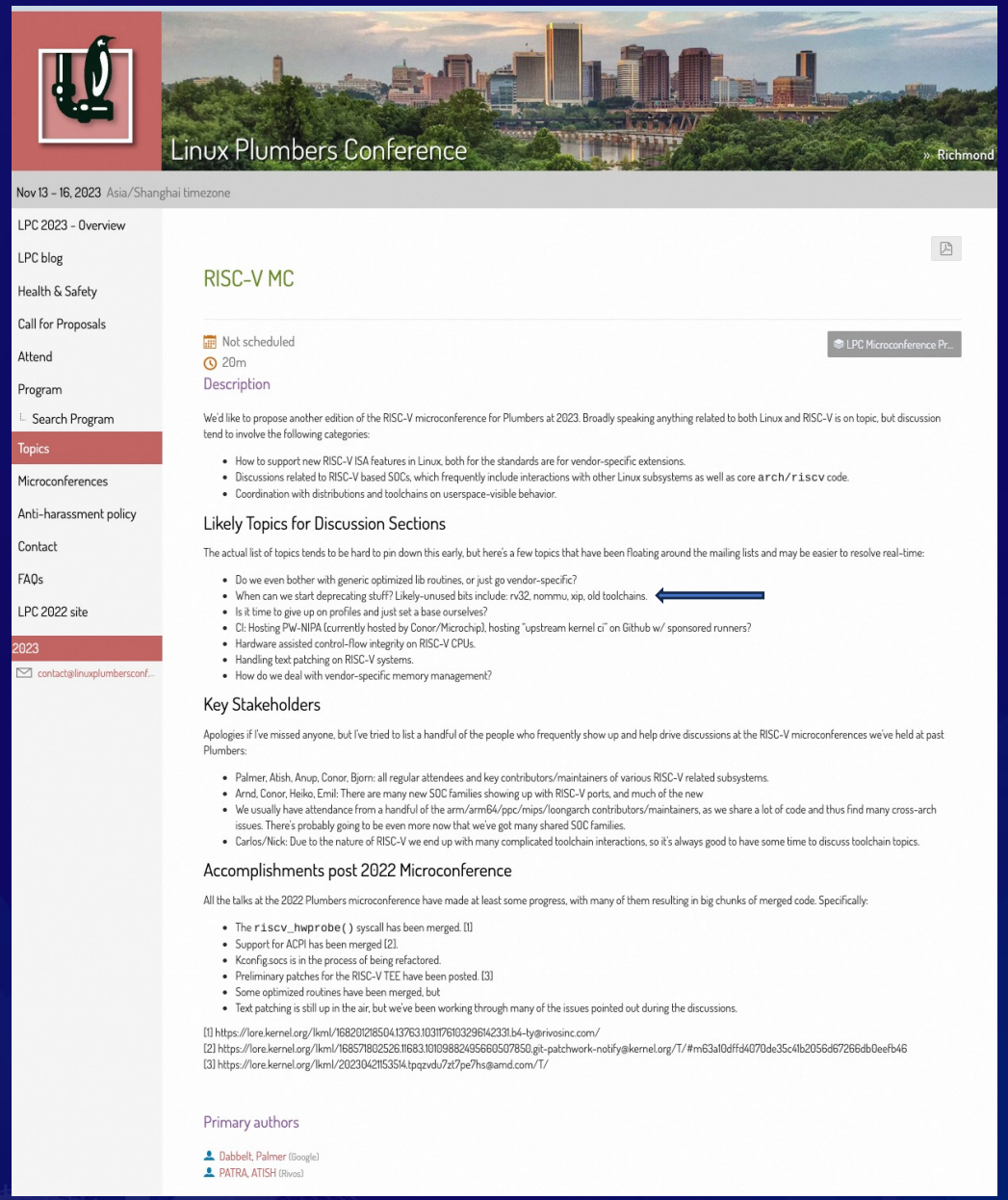
RV - RISC-V

A - application processors running rich operating systems

LPC 2023 RISC-V MC (把删除 rv32 当作一项议题)
Likely Topics for Discussion Sections

...

- When can we start deprecating stuff? Likely-unused bits include: **rv32**, nommu, xip, old toolchains.



32位指令架构的芯片面积小?

"Compared to Cortex-A35, the Cortex-A32 offers same 32-bit performance but consumes 10% less power and has a 13% smaller core."

https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/introducing-cortex-a32-arm-s-smallest-lowest-power-armv8-a-processor-for-next-generation-32-bit-embedded-applications

- A32 相比 A35，通用寄存器数量也从 32 个减至 16 个 (RISC-V 32/64位 寄存器数量相同)
- A32 相比 A35，少了16个 128 位SIMD寄存器 (同上)
- A35 作为 32位 + 64位的混合架构，额外包含了 A32 的所有控制，运算模块

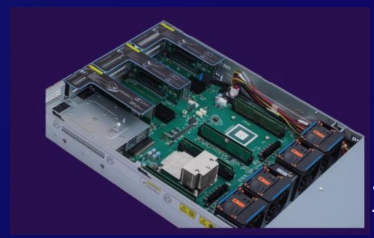
以上三点均不涉及位宽调整，那么 64 位到 32 位，在 13% 中占多少呢？姑且对折：6.5%，再加上 L2 cache，和各种 IP 后，按 AP 类芯片 CPU 核占 10% 面积算，32位架构带来的芯片面积收益约 0.65%

新32位内核相对老32位的巨大优势，使得芯片厂愿意升级到纯64位架构，这种潮流会给市场带来景气和繁荣。

挣脱32位的桎梏

(s 代表内核态，u 代表用户态)

- 低端：s64ilp32 + u64ilp32 (新32位内核)
- 中端：s64lp64 + u64ilp32
- 高端：s64lp64 + u64ilp32/u64lp64 (用户态，32位与64位的混合部署)



单核8MB ... 单核64MB ... 128核256GB ... 超级计算机

RV64 ISA + Linux

- 软件生态，无缝衔接
- 产品迭代，从容淡定
- 人才培养，循序渐进

fedora remix

即将推出全球首款新32位发行版
s64ilp32 + u64ilp32

https://fedoraproject.org/wiki/Architectures/RISC-V/64ILP32

Find More

Xuantie @
GitHub

挣脱32位的桎梏