RISC-V基础指令集在数据中心场景的评估和展望

侯俊杰 Hou Junjie - ByteDance

houjunjie.code@bytedance.com

ByteDance 字节跳动

CONTENTS

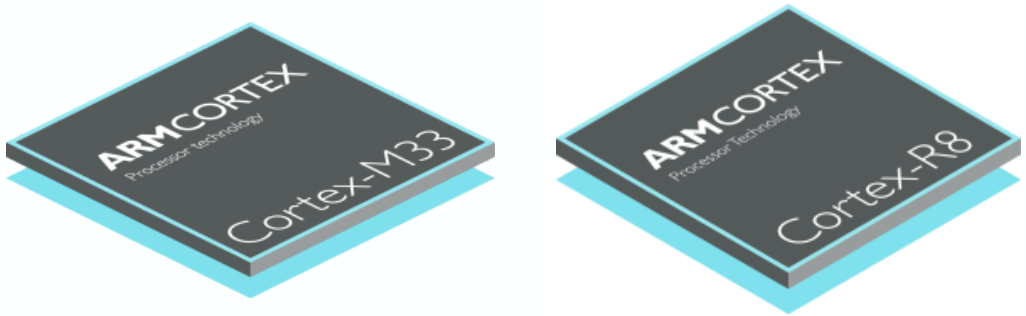
目录

- RISC-V指令集在应用之初主要面向嵌入式等低功耗应用
- 随着近些年RISC-V处理器在越来越多应用场景的成功，RISC-V逐渐进入数据中心等高性能场景

嵌入式/实时CPU

目标市场： 微控制器、嵌入式系统、实时控制系统（汽车、工业）、物联网节点。

- 低功耗
- 实时响应
- 确定性执行
- 小面积/低成本

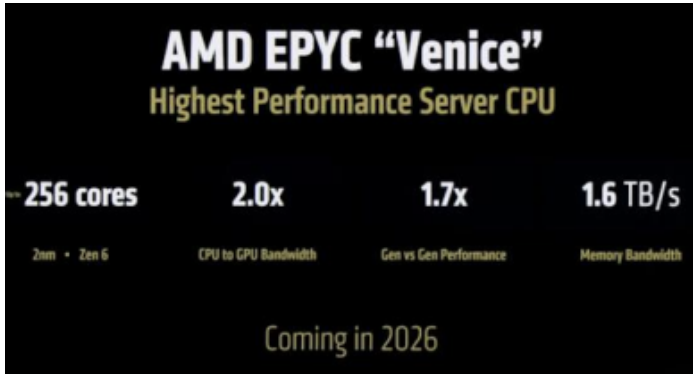


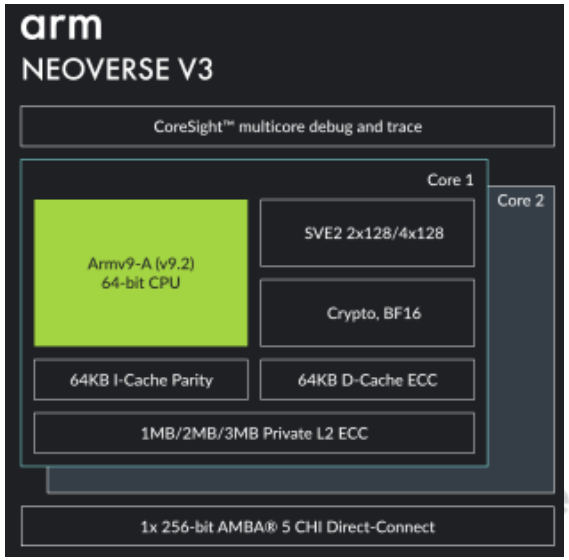
高性能/服务器 CPU（如数据中心、云计算处理器）

目标市场： 云服务器、数据中心、5G基础设施、高性能网络、边缘服务器。

极致追求性能、吞吐量、可扩展性、能效和特定工作负载优化（如云、网络、HPC）

- 高性能
- 高吞吐量
- 多核并行
- 虚拟化等高级特性





● 硬件需求差异

	嵌入式/实时CPU	高性能/服务器CPU
实时行	严格确定性执行 (很少或不支持投机执行)	允许非确定性优化 (深度投机，乱序执行)
功耗/性能	功耗要求较高，一般顺序执行或低乱序	更关注性能，乱序多发射
地址转换	简单或无MMU	复杂TLB结构，支持大页和多级页表
缓存层次	单级缓存或无缓存	复杂多级别缓存层次
核心数	单核或N核	128-256+
异常处理	简化的异常处理机制	复杂的精确异常和中断处理
虚拟化	无	支持容器等方案隔离应用

● 程序Code差异

特性	嵌入式/实时CPU	高性能/服务器CPU
程序CodeSize	小型指令缓存或无缓存，极小化代码体积 (存储成本敏感)	大型多级指令缓存，允许较大代码体积 (存储资源充足)
代码布局	紧凑布局，减少跳转开销	缓存友好型布局，提高指令获取效率
函数调用约定	最小化寄存器保存，减少栈操作	优化寄存器使用，支持复杂参数传递
对齐要求	严格对齐访问，减少硬件复杂度	支持高效非对齐访问

● 指令需求差异

特性	低功耗/嵌入式CPU	高性能/服务器CPU
指令长度 (如C扩展)	16位压缩指令优先，减少代码大小	代码大小影响很小，32位标准指令可以优化解码效率
Dynamic指令数	最小化动态指令数 (降低功耗，满足实时性)	允许一定动态指令数增加 (通过并行和缓存掩盖延迟)
高性能复杂指令	避免复杂指令 (简化流水线，降低功耗)	高频pattern复合/加速指令，提升单线程性能
控制流指令	简单条件分支，固定跳转范围	Code Footprint较大，控制流逻辑复杂，长距离跳转有需求
高级系统功能扩展	基本功能，最小实现	支持Server Platform/RVA23等完备扩展

CONTENTS

目录

应用/服务器CPU重要综合测试Benchmark: **SpecCPU2017**

- SpecCPU2017主要衡量**CPU算力、内存子系**统效率
- 与数据中心业务有部分重合，针对**CPU Core**有一定表征度

缺乏对**I/O、网络、存储、虚拟化、分布式计算**的表征能力

数据中心业务	有关联度SpecCPU子项
视频处理	525.x264_r (x264)
微服务	557.xz_r (数据压缩)
数据库	523.xalancbmk_r (XML解析)
	500.perlbench_r (脚本处理)
大数据处理	519.lbm_r (粒子动力学) 访存密集型，模拟Shuffle内存压力
搜索/推荐	531.deepsjeng_r(AI 搜索)
	541.leela_r(AI推理)

指令pattern	指令序列	指令pattern占比	指令pattern	指令序列	指令pattern占比
scaled_index_load	sh1/2/3add rd0 rs1 rs2 ld rd1 0(rd0)	0.638530%	scaled_index_store	sh1/2/3add rd0 rs1 rs2 st rd1 0(rd0)	0.017296%
sd_pair	sd rs2, offset+8(rs1) sd rs3, offset(rs1)	0.432830%	index_store	add rd0, rs1, rs2 st rd1, 0(rd0)	0.011010%
ld_pair	ld rd1, offset+8(rs1) ld rd2, offset(rs1)	0.395080%	lui_load_static	lui rd0, imm[31:12] ld rd1, imm[11:0](rd)	0.005348%
store_pre_incr	addi rs2, rs2, imm1 sd rs1, imm2(rs2)	0.364730%	auipc_genimm	auipc rd0, imm1[31:12] addi rd1, rd0, imm2[11:0]	0.001534%
loop_br	addi[w]/subi[w] rs1,imm br rs1 rs2	0.341770%	logic_extract_lsb	<logic op> r1 r2 r3 andi r1, r1, imm	0.001378%
index_load	add rd0, rs1, rs2 ld rd1, 0(rd0)	0.259870%	arith_zext	add[i]/sub[i] rd1 rs2 rs3 zext.h rd1 rd1	0.001054%
lui_genimm	lui rd, imm[31:12] addiw/addi rd, rd, imm[11:0]	0.259240%	auipc_load_static	lui rd0, imm[31:12] ld rd1, imm[11:0](rd)	0.000395%
br_imm	addi rd x0 imm br rs1 rd	0.171350%	get_second_byte	srli r1, r0, 8 andi r1, r1, 255 或者 slli r1 r0, 48 srli r1, r1, 56	0.000003%
long_jump	lui rd0, symbol[31:12] jalr rd1, symbol[11:0](rd0)	0.140330%	shift right by 29/30/31/32 and add	slli r1, r0, 32/31/30/29 srli r1, r1, 32/31/30/29	0.000001%
cmov	b.cond rs1, rs2, offset mv rd, rs3	0.096443%	logic operation and extract its lower 16 bits	<logic op> r1 r2 r3 zext.h r1, r1	0.000001%
load_pre_incr	addi rd0, rs1, imm1 ld rd1, rd0, imm2	0.076569%	bytemask add[w] and extract its lower bits	add[w] r1 r2 r3 andi r1, r1, mask	0.000000%
shift_left_by_4/5_and_add	slli r1, r0, 4/5 add r1, r1, r2	0.027995%	signed 16bit add/sub 等价于16bit无符号加减法	add[i]/sub[i] rd1 rs2 rs3 zext.h rd1 rd1	0.000000%
clear_upper_48_bits	slli r1, r0, 48 srli r1, r1, 48	0.020080%			

高频指令pattern SpecCPU 2017 Int(LLVM)频度Profiling (各子项占比数据Geomean)

SpecCPU2017分析的**高频基础指令Pattern** 与大部分业务上频度较为一致

- Load/Store Pair Pattern在大部分目标业务上为Top 5
- (Scaled)Index Load/Store Pattern在大部分目标业务上为Top 5
- Immediate Branch Pattern在部分业务为Top10
- Loop Branch在部分业务上为Top10

……

针对基础指令的分析，我们以SpecCPU2017作为目标测试和分析用例

SpecCPU作为开源Benchmark，更适合分析数据的直接对比和复现

CONTENTS

目录

Load/Store Operations

- **Load/Store Pair**

Main Scenarios

- **Stack Push/Pop**
- **Memcpy operations**

```
1786984    ld    s3, 488(sp);ld   s4, 480(sp)
1786988    ld    s5, 472(sp);ld   s6, 464(sp)
1786992    ld    s7, 456(sp);ld   s8, 448(sp)
1786996    ld    s9, 440(sp);ld   s10, 432(sp)
1787090    ld    a0, 16(a0);ld    a0, 8(a0)
1787494    ld    ra, 40(sp);ld    s0, 32(sp)
1787498    ld    s1, 24(sp);ld    s2, 16(sp)
```

Patterns Snippet

- SpecCPU2017 **静态CodeSize**降低0.52%

	llvm19-riscv64	llvm19-riscv64(ldp/stp-off)	Decreased Proportion
500-perlbench_r_base	2307886	2380086	**3.03%**
502-cpugcc_r_base	7668420	7658236	-0.13%
505-mcf_r_base	715402	715402	0.00%
520-omnetpp_r_base	2261502	2284106	**0.99%**
523-cpuxalan_r_base	3255348	3281852	**0.81%**
525-x264_r_base	1103812	1106124	0.21%
525-imagevalidate_525_base	664654	664958	0.05%
525-ldecod_r_base	1068372	1068708	0.03%
531-deepsjeng_r_base	751642	752558	0.12%
541-leela_r_base	1335156	1336544	0.10%
548-exchange2_r_base	736212	736212	0.00%
557-xz_r_base	717500	719008	0.21%
Sum	22585906	22703794	0.52%

Codesize Benefits (.text segment Bytes)

- SpecCPU2017 动态**指令数**降低3.55%

	llvm19-riscv64	llvm19-riscv64(ldp/stp-off)	Decreased Proportion
500	2.80E+12	3.10E+12	**9.67%**
502	1.05E+12	1.13E+12	**6.78%**
505	6.54E+11	6.57E+11	0.42%
520	1.04E+12	1.09E+12	**5.22%**
523	1.08E+12	1.10E+12	1.54%
525	3.86E+12	3.89E+12	0.83%
531	1.73E+12	1.76E+12	1.97%
541	1.96E+12	2.11E+12	**6.70%**
548	2.06E+12	2.06E+12	0.00%
557	1.77E+12	1.77E+12	0.03%
Sum	1.80E+13	1.87E+13	3.55%

Dynamic Instruction Count Benefits

- **Addressing Mode**

相比ARM和X86，RISC-V寻址模式单一，仅**支持base+displacement，其他类型需要多条指令组合实现**

- **(Scaled) Indexed Load/Store**（**add/shadd rd0, rs1, rs2 (1/2/3)** + **ld rd1, imm(rd0)**或store)
- **Pre/Post Increment Load/Store** (**addi rd0, rs1, imm1** + **ld rd1, rd0, imm2**或store)
- Absolute Load (**lui rd0, imm1[31:12] + ld rd1, imm2(rd0)**)
- PC-relative Load (**auipc rd0, imm1[31:12] + ld rd1, imm2(rd0)**)

Common addressing mode name [8]	Addressing components	x86-64 [9]	Aarch64 [10]	RISC-V [2]
Indirect	[base]	[rax]	[x0]	(a0)
Relative	[base+displacement]	[rax+10]	[x0, 10]	10(a0)
Indexed	[base+index]	[rax+rdx]	[x0, x1]	N/A
Indexed-Relative	[base+index+disp.]	[rax+rdx+8]	N/A	N/A
Scaled Index	[base+index*scale]	[rax+rdx*4]	[x0, x1, lsl 2]	N/A
Scaled Index Relative	[base+index*scale+disp.]	[rax+rdx*4+8]	N/A	N/A
Absolute	[displacement]	[0x8000]	N/A	N/A
PC-Relative	[pc+displacement]	[rip+0x1234]	0x8000	0x8000

(Scaled) Indexed Load/Store (add/shadd rd0, rs1, rs2 (1/2/3) + ld rd1, imm(rd0)或store)

- **Register Pressure Benefits:**

Register **Spills decrease by 2.0%** and **reloads decrease by 1.2%**(SpecCPU 2017)

- SpecCPU2017 **静态CodeSize**降低0.80%

	llvm19-riscv64	llvm19-riscv64(zilsx-off)	Decreased Proportion
500-perlbench_r_base	2300350	2380086	**3.35%**
502-cpugcc_r_base	7631460	7658236	0.35%
505-mcf_r_base	714902	715402	0.07%
520-omnetpp_r_base	2260182	2284106	**1.05%**
523-cpuxalan_r_base	3250112	3281852	**0.97%**
525-x264_r_base	1100968	1106124	0.47%
525-imagevalidate_525_base	664586	664958	0.06%
525-ldecod_r_base	1061032	1068708	**0.72%**
531-deepsjeng_r_base	751402	752558	0.15%
541-leela_r_base	1333844	1336544	0.20%
548-exchange2_r_base	736212	736212	0.00%
557-xz_r_base	716424	719008	0.36%
Sum	22521474	22703794	**0.80%**

Codesize Benefits (.text segment Bytes)

- SpecCPU2017 动态**指令数**降低3.55%

	llvm19-riscv64	llvm19-riscv64(zilsx-off)	Decreased Proportion
500	2.94E+12	3.10E+12	**5.25%**
502	1.11E+12	1.13E+12	1.52%
505	6.58E+11	6.57E+11	-0.20%
520	1.05E+12	1.09E+12	**4.32%**
523	1.09E+12	1.10E+12	0.67%
525	3.87E+12	3.89E+12	0.59%
531	1.71E+12	1.76E+12	3.20%
541	1.89E+12	2.11E+12	**10.12%**
548	2.06E+12	2.06E+12	0.00%
557	1.63E+12	1.77E+12	**7.72%**
Sum	1.80E+13	1.87E+13	3.55%

Dynamic Instruction Count Benefits

Branch and Jump

- **Branch with Immediate**

```c
int f(int i)
{
        if (i > 5)
                i += 11;
        else
                i += 7;

        return i;
}
```

```
        st x8 [sp]
        li x8 #5
        bge .target x5 x8
        ...

.target
        ...
        ld x8 [sp]
```

- bne (32.61%) and beq (21.04%) as dominant operators

	Static		Dynamic	
Comparison	Frequency	Proportion (%)	Frequency	Proportion (%)
beq	45600	32.37%	3.28E+10	21.04%
bne	57565	40.86%	5.09E+	32.61%
bltu	21091	14.97%	3.98E+10	25.50%
blt	5142	3.65%	1.73E+10	11.06%
bgeu	8509	6.04%	1.18E+10	7.59%
bge	2984	2.12%	3.44E+09	2.20%

Comparison Operators

- **the average 7.60% (some benchmarks are more than 15%)** of conditional branches are with immediate values.

	A	B	C	F
1	Workload	Number of Branch with Immediate patterns	Number of Conditional Branches	Branch with Immediates as a proportion of Conditional Branches
2	**Overall**	1.56E+11	2.05E+12	7.60%
3	500.perlbench_r.checkspam.2500.5.25.11.150.1.1.1.1	2.19E+10	1.36E+11	16.13%
4	500.perlbench_r.diffmail.4.800.10.17.19.300	2.02E+10	9.52E+10	21.24%
5	500.perlbench_r.splitmail.6400.12.26.16.100.0	1.51E+10	9.42E+10	16.08%
6	502.gcc_r.gcc-pp.opts-O2_-finline-limit_36000_-fpic	7.00E+09	3.61E+10	19.39%
7	502.gcc_r.gcc-pp.opts-O3_-finline-limit_0_-fif-conversion_-fif-conversion2	5.64E+09	3.00E+10	18.77%
8	502.gcc_r.gcc-smaller.opts-O3_-fipa-pta	5.22E+09	3.97E+10	13.14%
9	502.gcc_r.ref32.opts-O3_-fselective-scheduling_-fselective-scheduling2	5.89E+09	3.70E+10	15.92%
10	502.gcc_r.ref32.opts-O5	5.09E+09	2.98E+10	17.10%
11	505.mcf_r.inp	2.53E+09	1.47E+11	1.72%
12	520.omnetpp_r.omnetpp.General-0	1.19E+10	1.35E+11	8.84%
13	523.xalancbmk_r.ref-t5	1.64E+09	3.00E+11	0.55%
14	525.x264_r.run_000-1000_x264_r_pass1	5.45E+08	1.60E+10	3.40%
15	525.x264_r.run_000-1000_x264_r_pass2	1.42E+09	5.67E+10	2.51%
16	525.x264_r.run_0500-1250_x264_r	1.44E+09	5.77E+10	2.50%
17	531.deepsjeng_r.ref	2.10E+10	1.41E+11	14.95%
18	541.leela_r.ref	1.89E+10	1.78E+11	10.58%
19	548.exchange2_r.exchange2	3.02E+09	2.56E+11	1.18%
20	557.xz_r.cld.tar-160-6	1.96E+09	3.47E+10	5.63%
21	557.xz_r.cpu2006docs.tar-250-6e	2.91E+09	1.69E+11	1.72%
22	557.xz_r.input.combined-250-7	2.73E+09	6.45E+10	4.24%

Branch with Immediate Pattern Frequency

- **Branch with Immediate**

Register Spill/Reload Benefits

SpecCPU 2017

Spills decrease by 1.82% and reloads decrease by 2.24%

llvm-test-suite

Spills decrease by 1.82% and reloads decrease by 1.42%

	llvm21-riscv64	llvm21-riscv64(zibimm-off)	Decreased Proportion
500.perlbench_r.checkspam.2500.5.25.11.150.1.1.1.1	3.29E+12	3.29E+12	-0.04%
502.gcc_r.gcc-pp.opts-O2_-finline-limit_36000_-fpic	1.19E+12	1.21E+12	1.38%
505.mcf_r.inp	9.18E+11	9.18E+11	0.02%
520.omnetpp_r.omnetpp.General-0	1.11E+12	1.11E+12	0.34%
523.xalancbmk_r.ref-t5	1.25E+12	1.25E+12	0.17%
525.x264_r.run_000-1000_x264_r_pass1	4.43E+12	4.43E+12	0.03%
531.deepsjeng_r.ref	2.01E+12	2.03E+12	0.88%
541.leela_r.ref	2.53E+12	2.55E+12	0.60%
548.exchange2_r.exchange2	1.94E+12	1.94E+12	0.00%
557.xz_r.cld.tar-160-6	1.85E+12	1.85E+12	0.09%
sum	2.05E+13	2.06E+13	0.28%

- **Other Branch/Jump Patterns**

Codesize Benefits (.text segment Bytes)

  - **Loop Branch (addi rs2, rs1, imm1 + branch rs1, rs2, .target)**
  - Long Jump (**auipc rd0, imm1[31:12] + jalr rd1, imm2(rd0)**)
  - Jump Global (**lui rd0, imm1[31:12] + jalr rd1, imm2(rd0)**)
  - Conditional MOV (**b.cond rs1, rs2, offset + mv rd, rs3**)

- **Full Immediate Operation**

  Generate Immediate **(lui rd0, imm1[31:12] + addi rd1, rd0, imm2[11:0])**

  - PC-Based Generate Immediate **(auipc rd0, imm1[31:12] + addi rd1, rd0, imm2[11:0])**

- **Logic Operation**

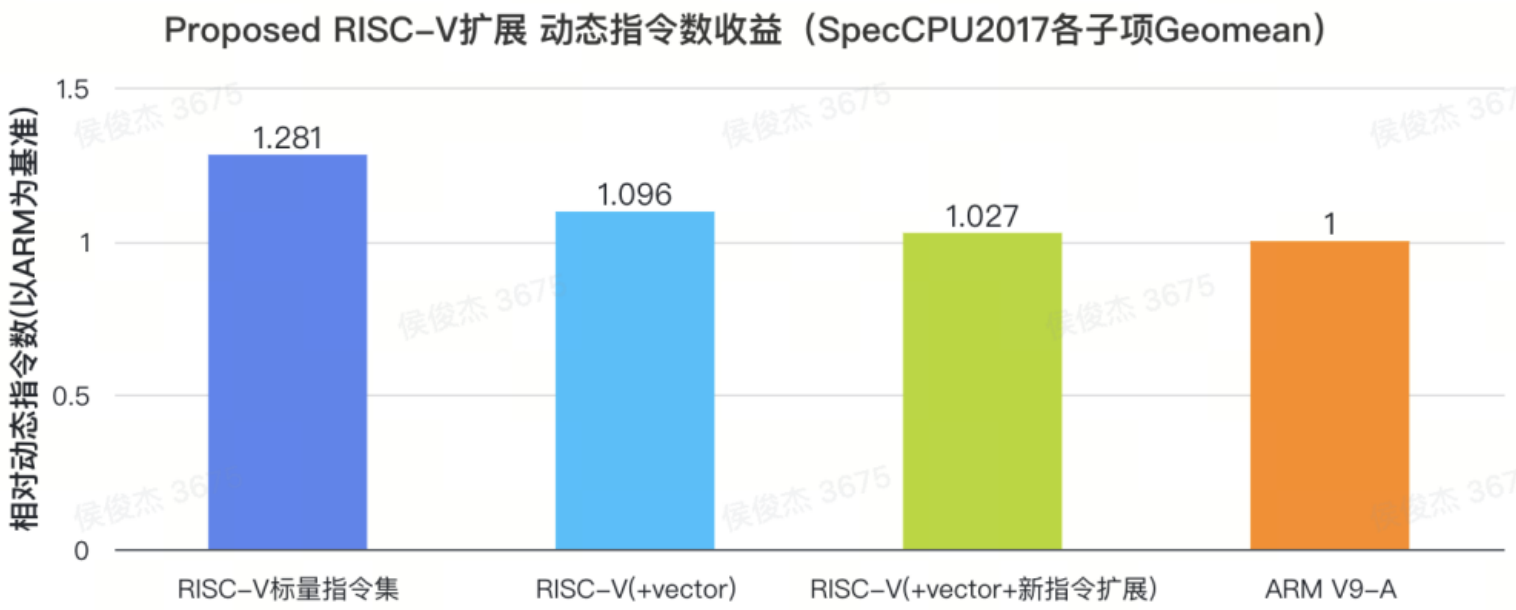
  Shift and Add **(slli rd1, rs1 imm + add rd1, rd1, rs2)**

  Clear Bits **(slli rd1 rs1, imm + srli rd1, rd1, imm)**

  Extract Bits **(<logic op> rd1 rs1 rs2 + andi rd1, rd1, imm)**

  Arithmetic and Extend**(add[i]/sub[i] rd1 rs2 rs3 + zext.h rd1 rd1)**

- 使能向量和新指令扩展，SpecCPU2017 动态**指令数RISC-V 与ARM间差异3%以内**



备注：

标量指令集:

-march=rv64imafdc_zba_zbb_zbc_zbs_zicond_zfa

编译工具链：LLVM19

新指令扩展:

（Load/Store Pair, Index Load/Store, Immediate Branch）

Optimize_Flags: -O3 -funroll-loops -mabi=lp64d -flto

CONTENTS

目录

扩展指令（RISC-V64）Proposal社区状态@侯俊杰@汪鹏程

新指令扩展（RV64）

ByteDance扩展Proposals	社区状态
Branch with Immediate	已接受Fast Track
Load/Store Pair	当前社区建议通过微架构Fusion实现
(Scaled) Indexed Load/Store	认可分析结果和支持必要性，预计后续Fast Track
Pre-Post Index Load/Store	社区讨论中
Integer multiply add (madd)	待讨论
branch.far(128k conditional branch)	待讨论
...	

微架构指令Fusion

- 指令融合技术可以在硬件层面将连续的多条指令合并执行，从而提高执行效率，减少实际执行的指令数
- 定义新指令无法编码/无明显指令数/CodeSize/寄存器压力收益pattern，通过微架构fusion实现作为性能优化手段

展望

- 核心高频操作与X86/ARM间的差异补齐
- 当前新指令扩展明显受限于编码空间，编码空间问题需早做规划（新增扩展只支持部分操作类型）
- C扩展带来codesize的收益对数据中心场景不敏感，是否考虑针对数据中心场景排除C扩展，扩充针对高性能场景的扩展指令
- 数据中心通用业务场景的加速指令支持， 避免vendor自定义导致生态的碎片化
- 数据中心业务性能与成本高度关联，针对业务的指令扩展是相比其他架构的关键竞争力

未来期望与各RISC-V厂商共同推动RISC-V在数据中心等高性能应用场景的完善和成熟

THANKS

ByteDance字节跳动