大会特邀报告三：指令系统的自主与兼容

特邀报告嘉宾：胡伟武 研究员

胡伟武，男，1968 年11 月出生于浙江永康，1986 年中学毕业于永康一中，1991 年7 月大学毕业于中国科学技术大学计算机系，随后免试进入中科院计算所直接攻读博士学位，师从著名计算机专家夏培肃院士，1996 年3 月获工学博士学位，博士论文被评为全国百篇优秀论文。现任龙芯中科技术有限公司董事长、总经理，中科院计算所总工程师、研究员、博士生导师，第十一届全国青联常委，第十一届全国人大代表，党的十八大、十九大代表。 胡伟武研究员2001 年起投身于龙芯处理器的研制工作。先后主持完成了我国第一个通用处理器龙芯1 号、第一个64 位通用处理器龙芯2 号、第一个四核处理器龙芯3 号的研制，使我国处理器研制达到世界先进水平。目前，龙芯处理器已经形成系列产品，广泛应用于军工、党政办公、工控、嵌入式等领域，为国家安全和自主信息产业发展做出了贡献。

指令系统的自主和兼容

胡伟武

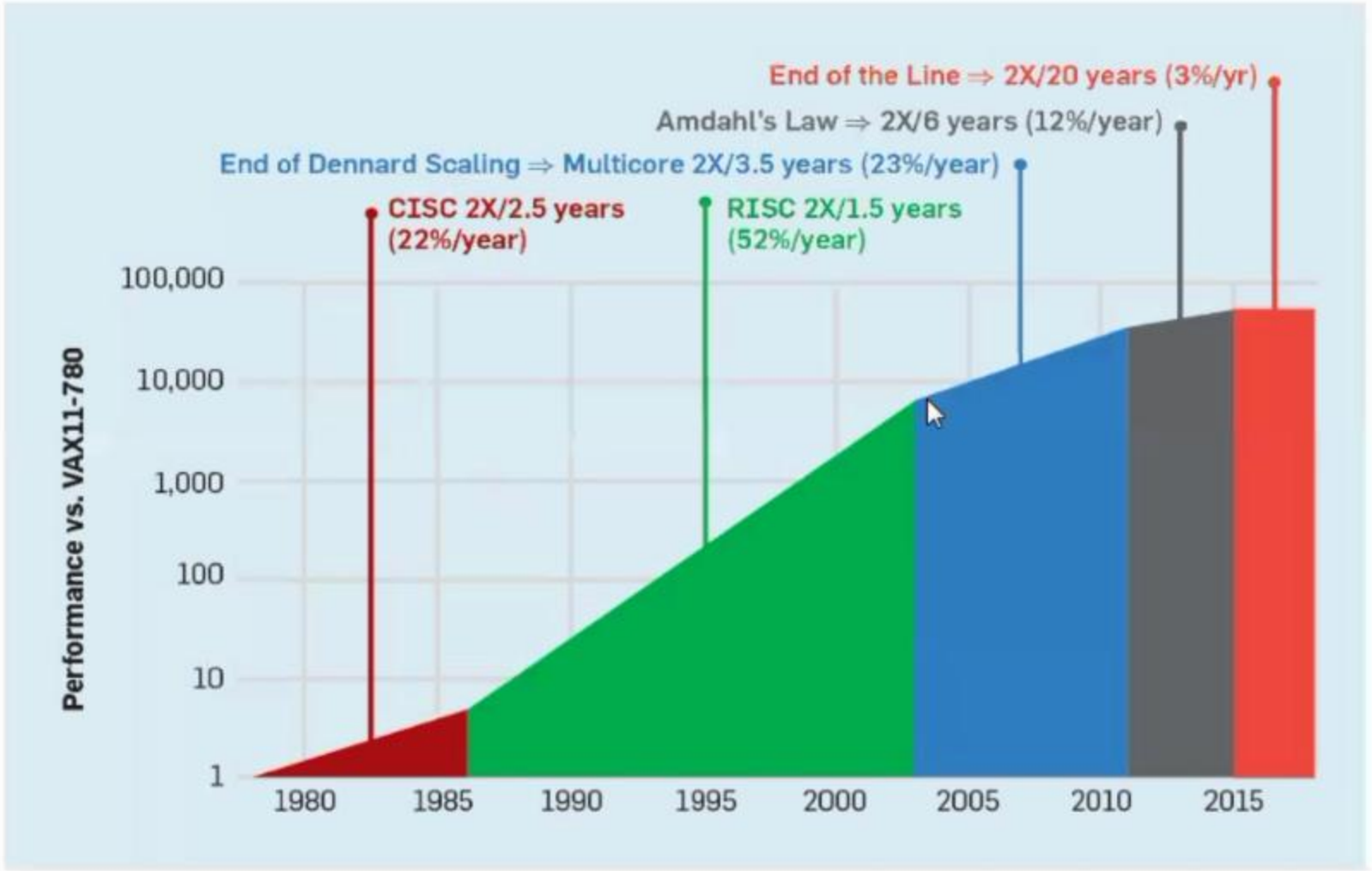
龙芯中科技术有限公司

中国科学院计算技术研究所

建立自主信息产业体系成为国家战略

- 美国决心限制我国信息产业往高端发展，我国决心建立自主信息产业体系
  - 已成为美国国民共识，与谁当美国总统无关：一些列断供事件
  - “构建独立于Wintel体系和AA体系的安全可控的信息技术体系和产业生态”
- CPU性能“补课”基本完成
  - 难的是单核通用处理性能
  - 28nm的龙芯3A4000单核SPEC CPU2006分值20分，预计14nm的3A5000单核分值可达30分以上
- 自主化应用带动自主CPU快速发展
  - 应用范围不断扩大：党政、轨交、电力、金融、教育……
  - 从信息安全到产业发展：技术快速迭代、产业链深度融合、资本高度活跃
  - 引进技术、自主研发并存的多款CPU：龙芯发挥自主性强、开放性好的优势实现超常规发展
- 自主CPU两大“卡脖子”问题：指令系统架构、生产工艺受制于人

CPU通用处理性能（SPECint）趋于极限



- 市场主流产品的单核SPEC INT2006分值20-40分（GCC编译）

指令系统是绕不过去的话题

- 自主与兼容指令系统的长期争论
  - 从“核高基”立项起展开激烈的自主CPU指令系统争论
- 兼容的好处：软件生态
  - X86的Windows和ARM的Android
  - 即使是开源社区也是X86的软件最完善，ARM次之
- 兼容的弊端：受制于人
  - X86不授权，ARM严格授权，MIPS授权比较开放（长期、自主加指令）
  - 兼容阻碍以操作系统为代表的自主基础软件的发展
- 我国不可能基于国外指令系统建设自主生态
  - 基于X86不可能打造自主生态；基于ARM不可能打造自主生态
  - 基于MIPS不可能打造自主生态；基于RISC-V不可能打造自主生态

能否能在自主指令系统上实现兼容？

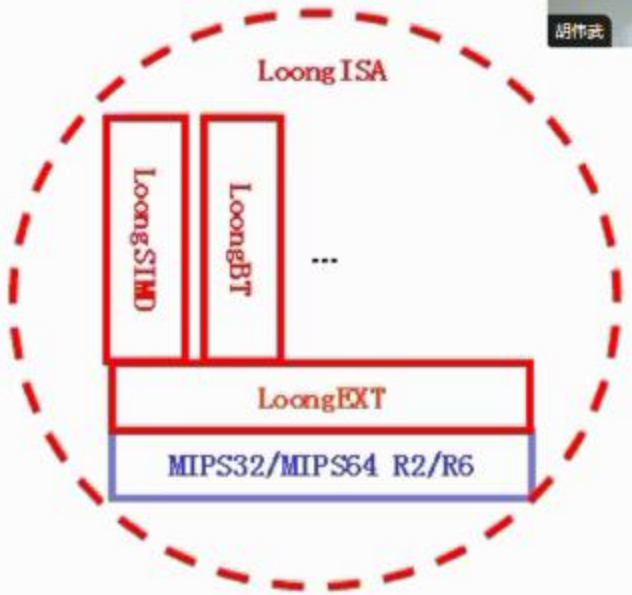
- 基础软件（BIOS、内核、编译器/汇编器）
  - BIOS（PMON/UEFI）、内核（Linux/VxWorks）：工作量不大
  - 汇编器及编译器（GCC、LLVM、GOLANG）：工作量可控
  - 整个操作系统重新迁移和编译工作量不大，支持直接把MIPS汇编语言编译成自主指令系统
- 动态翻译虚拟机（ Java、JavaScript、.NET ）
  - Java、JavaScript、.NET：均由龙芯完成迁移
  - Java、JS、.NET应用不用改，可以直接跑
- 二进制翻译（X86、ARM、MIPS）
  - 已有的龙芯应用不用重新迁移，直接在龙芯上跑
  - 在Android上直接运行 ARM应用，运行Windows及其应用
  - Qemu都可以实现，关键是提高运行效率（跨指令系统一般不到5%）

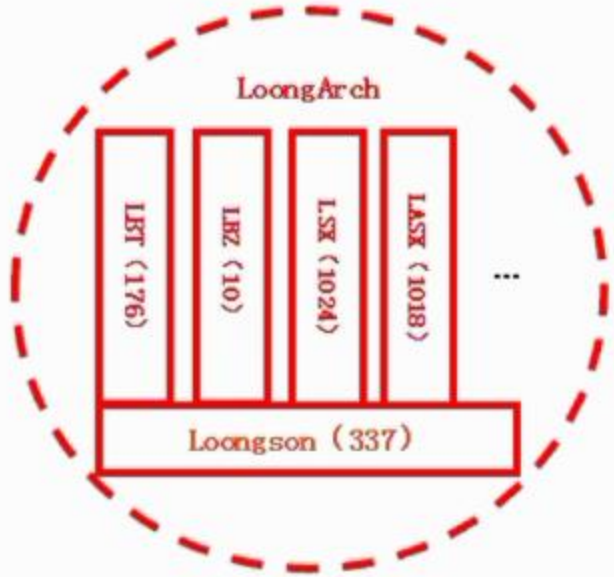
考虑兼容需求的自主指令系统

- 基本要求
  - 先进性：积极吸收近年来指令集发展的先进技术成果，适度摒弃一些“过时”的技术特征
  - 兼容性：融合X86、MIPS、ARM指令系统的主要特点，高效支持二进制翻译
  - 扩展性：指令槽留有余地，利于今后的持续演进
- 核心态的考虑（内核专用）
  - 内核（Linux/VxWorks）重新迁移，不用太多考虑与已有的兼容
  - 硬件支持两级地址翻译：X86到Loongarch + 虚地址到物理地址
  - 地址空间、中断处理等支持操作系统跨主板和对升级后的CPU兼容（只有Wintel做到）
- 用户态的考虑
  - 功能上充分考虑MIPS、X86、ARM、RISC-V的主要特征，绝大多数指令1对1或1对2翻译
  - 如X86的EFLAG支持、RISC-V的原子同步指令支持
  - ABI的实现支持X86、MIPS系统调用兼容，支持MIPS汇编码直接翻译成Loongarch二进制

龙芯指令系统

- LoongISA：基于MIPS架构的UDI进行扩展
  - MIPS： 279条基础+900条SIMD+37条加解密
  - 基础扩展LoongEXT（共186条）
  - 二进制翻译扩展LoongBT（共157条）
  - 向量指令扩展LoongSIMD（共477条，不含256位扩展）
- LoongArch：龙芯自主架构（2500+条指令）
  - “充分考虑兼容需求的自主指令系统”
  - 基础指令337条
  - 虚拟机扩展10条
  - 二进制翻译扩展（X86、ARM等）176条
  - 128位向量扩展1024条
  - 256位向量扩展1018条



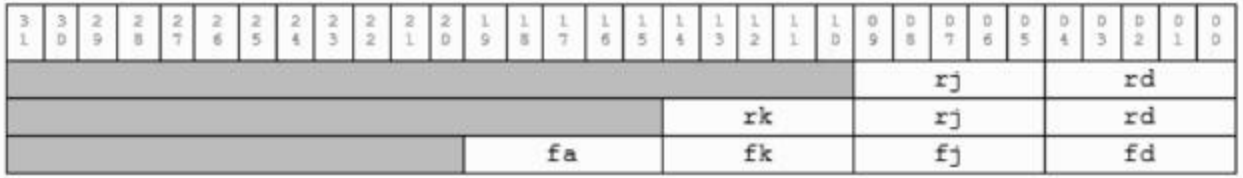


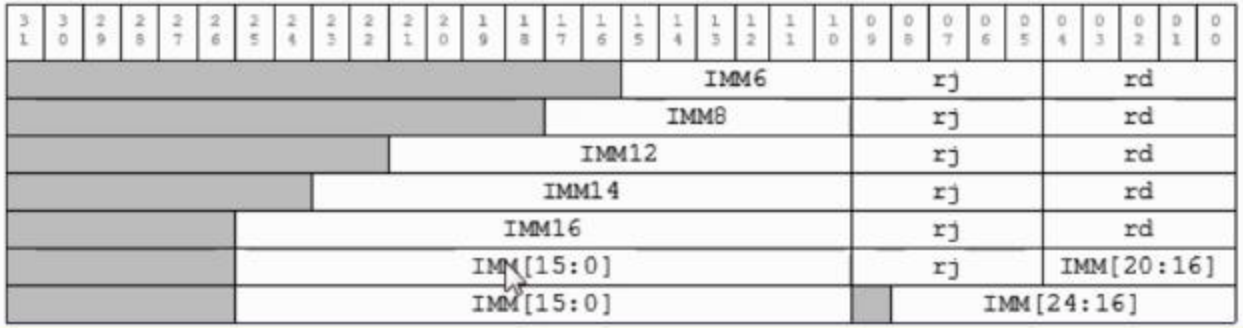
MIPS与Loongarch格式

- MIPS的三种格式

R-type	OP(6)	RS1(5)	RS2(5)	RD(5)	SA(5)	OPX(6)
I-type	OP(6)	RS(5)	RD(5)	Immediate		
J-type	OP(6)	target				

- Loongarch的3种无立即数格式和7种有立即数格式（精打细算）
  - 尽量满足部分指令长立即数指令要求
  - 节省指令槽，留有一半一级指令槽





Loongarch特点

- 用户态提升指令易用性与执行效率
  - 保持“典型”RISC的特点： 32位定长指令、32个通用寄存器、32个浮点/向量寄存器
  - 取消转移指令延迟槽，直接跳转指令的目标地址相对PC计算，增加相对转移偏移量
  - 新增将PC作为源操作数的运算指令
- 核心态更适合现代操作系统

	MIPS	Loongarch
运行模式	用户态、监管态、核心态三级	PLV0~PLV3四级
计时系统	计时频率随处理器核频率变化	计时频率恒定
例外/中断	27种例外，多个例外共用入口 用EXL、ERL寄存器辅助处理嵌套例外	25种例外，每个例外独立入口 用PPLV、PIE和scratch寄存器辅助处理嵌套例外
内存管理	寻址空间固定分段，且与允许的运行模式、虚实映射规则紧密相关	单一平整（flat）寻址空间，支持所有空间均用页表映射
控制寄存器	最多2^8个，不支持域的原子修改 没有核外控制寄存器的规范	最多2^{14}个，支持域的原子修改 规范核外控制寄存器且使用独立的寻址空间

Loongarch对二进制翻译的支持

- 融合X86、ARM、MIPS、RISC-V指令系统的主要功能
  - 定义专门产生EFLAG结果的运算类指令加速X86、ARM架构下独有的EFLAG值模拟
  - 设计浮点寄存器的栈访问模式加速X87浮点协处理器指令的模拟
  - 实现RISC-V中丰富的同步指令
  - ……
- 对高效率二进制翻译过程的支持
  - 调整TLB结构使其能够直接用于二进制翻译过程中的访存操作的两级虚实地址映射，在系统级翻译中地址翻译（如把X86虚地址翻译成Loongarch的虚地址）占运行时间达30-40%
  - 设计便签寄存器、专用插桩转移指令等加速翻译中边角情况的处理
  - ……

X86二进制翻译举例

0		SUB	ECX	EDX		
1		JE	X86_target			

- EFLAG问题
  - X86通过EFLAG标志来表示运算结果并用于转移条件
  - MIPS没有EFLAG标志，需要40多条指令来模拟
  - 定义新的MIPS格式指令来产生EFLAG标志。

0.00		SUBU	Result	Recx	Redx	
0.01		SRL	Rsf	Result	31	/*SF=Result[31]*/
0.02		BEQ	Result	R0	L1	
0.03		ADD	Rzf	R0	R0	/*ZF=0*/
0.04		B	L2			
0.05		NOP				
0.06	L1:	ADDI	Rzf	R0	1	/*ZF=1*/
0.07	L2:	SRL	Rtmp1	Result	31	
0.08		SRL	Rtmp2	Redx	31	
0.09		SRL	Rtmp3	Recx	31	
0.10		BEQ	Rtmp1	Rtmp3	L3	
0.11		NOP				
0.12		BEQ	Rtmp2	Rtmp3	L3	
0.13		NOP				
0.14		ADDI	Rof	R0	1	/*OF=1*/
0.15		B	L4			
0.16		NOP				
0.17	L3:	ADD	Rof	R0	R0	/*OF=0*/
0.18	L4:	SRL	Rhigh2	Recx	16	
0.19		SRL	Rhigh1	Redx	16	
0.20		SUBU	Rtmp	Rhigh2	Rhigh1	
0.21		BEQ	Rtmp	R0	L5	
0.22		NOP				
0.23		BLTZ	Rtmp	L7		
0.24		NOP				
0.25		B	L6			
0.26		NOP				
0.27	L5:	SLL	Rlow2	Recx	16	
0.28		SRL	Rlow2	Rlow2	16	
0.29		SLL	Rlow1	Redx	16	
0.30		SRL	Rlow1	Rlow1	16	
0.31		SUBU	Rtmp	Rlow2	Rlow1	
0.32		BLTZ	Rtmp	L7		
0.33		NOP				
0.34	L6:	ADD	Rcf	R0	R0	/*CF=0*/
0.35		B	L8			
0.36		NOP				
0.37	L7:	ADDI	Rcf	R0	1	/*CF=1*/
0.38	L8	ADD	Recx	Result	R0	
1.00		BNE	Rzf	R0	MIPS_target	
1.01		NOP				

0.0		SUB	Result	Recx	Redx	/*Generating Sub result*/
0.1		X86SUB	Reflag	Recx	Redx	/*Generating EFLAGS*/
1.0		X86JE	Reflag	MIPS_target		/*Branch on EFLAGS*/

访存地址翻译和加速

X86指令

```
movl    0xc(%esi), %eax
```

QEMU softmmu：翻译成MIPS后的快速路径（TLB命中的情况，共16条指令）

```
lw          s1,24(s0)       # load %esi, s0是结构体CPUX86State的基地址，24是esi寄存器对应的偏移。
addiu       s2,s1,12        # 加偏移得到S2 = gva(guest虚拟地址)
ld          at,-32(s0)      # 开始访问软TLB，把gva翻译成host虚拟地址。取掩码
ld          t9,-24(s0)      # 软tlb表头
srl         t7, s2, 0x7     # gva对应的表偏移
and         t7,t7,at
daddu       t7,t7,t9
lwu         at,0(t7)          # 取软TLB表项内容
li          t9,-4093
ld          t8,16(t7)         # 软TLB存放gva和hva的偏移差值
and         t9,t9,s2
dext        a0,s2,0x0,0x20
bne         at,t9, not_hit  # 是否命中
daddu       a0,t8,a0        # gva -> hva (放a0寄存器)
lw          s2,0(a0)          # 取出0xc(%esi)的值
sw          s2,0(s0)          # 存到%eax
```

不命中的情况下将保存上下文后调用一个QEMU函数完成处理，代价在几百条指令以上

使用硬件TLB加速后的翻译

```
setmem <guest mem id>    #使用龙芯TLB扩展，给guest地址提供专门machine id空间，用前缀指令和普通访存区分
lw    0xc(S5), S2        #X86的ESI/EAX分别映射到S5/S2寄存器，若不命中，例外处理用软件TLB路径访问，更新TLB
b     1f                 #硬件加速访问成功则继续处理后续指令
<原来的softmmu处理代码>
1: <后续指令的翻译>
```

支持二进制翻译的开销

- 面积和延迟开销可忽略不计
- 通用CPU中80%面积在为运算器提供足够的数据和指令
  - 多级高速缓存（Cache）、转移猜测表、动态调度
  - 上述微结构的开销与指令系统基本无关
- 运算器中80%面积在浮点运算
  - 二进制翻译硬件支持主要涉及定点运算和访存地址运算

二进制翻译系统的特点

- 二进制翻译是体系结构翻译器
  - 在一种架构上运行为另一种架构生成的二进制代码
  - 不同体系结构的语义鸿沟：指令集、ABI、OS/工具链、应用层差异
  - 技术挑战：实现精确翻译复杂度高工作量大，性能损失不易控制
- 二进制翻译的技术途径
  - 进程级翻译（在龙芯Linux上运行EDA软件） vs. 系统级翻译（在龙芯上运行Windows 10）
  - 静态翻译（翻译完再运行） vs. 动态翻译（运行时翻译） vs. 动静态结合
- 技术发展给二进制翻译带来新机遇
  - 硬件资源极大丰富：晶体管过剩、CPU性能过剩
  - 虚拟机技术发展：二进制翻译本质上是一种跨指令系统的虚拟机

语言文化和体系结构的类比

- 实现语言级（不是单词级）的翻译
  - 定义新指令系统时要尽量缩小与其它指令系统的各方面差异
  - 繁体中文 => 简体中文；法文 => 英文；英文 => 中文
- 语言的通用翻译机正在成为现实
  - 计算性能的提高；算法的改进
- 体系结构通用翻译机？

语言文化	计算机体系结构
用户和商业支撑	用户和商业支撑
书籍、歌曲等	应用程序
语法、成语俚语、文化内核	ABI、函数库、OS
字、单词、发音	指令集

59种语言（含中文）实时互译，语种覆盖全球200个国家和地区。其中中英在线翻译能**实现0.5秒快速响应，翻译效果可媲美专业八级。**

二进制翻译的历史经验

- 向死而生
  - 大都是为了新架构铺路，新架构失败它消亡，成功它也消亡
  - IBM/HP/Intel/Apple/Transmeta/QualComm/Nvidia等都采用这个技术来协助推新架构
  - 70年代迄今一直在发展，数十种不同的翻译系统和无数的研究论文
- 二进制翻译的法律问题
  - 有不同的看法，美国等发达国家法律普遍认为不侵权
  - Transmeta赢了与Intel的官司

龙芯二进制翻译的持续积累

- 十余年近20位硕士博士毕业于相关课题
- 三代CPU迭代，不断完善对二进制翻译的支持
  - 3A1000、3A2000/3A3000、3A4000
  - 缩小指令语义差距的硬件支持：X86 eflags支持、浮点栈模式支持等
  - 地址翻译加速硬件把X86/ARM等的虚地址直接映射为Loongarch的物理地址
- 高效翻译引擎
  - 翻译X86应用：开源QEMU < 5%性能 vs. 龙芯引擎>60%性能
  - 翻译安卓ARM应用：流畅运行移动版WPS和美图秀秀等应用
- 某“备胎”课题顺利通过验收
  - 与ARM Cortex A系列可比的龙芯CPU核
  - 在Android上通过二进制翻译流畅运行ARM平台APP（如WPS）

龙芯二进制翻译系统LAT

- Loongson Architecture Translator
- 技术路线：动静结合、软硬结合、强化投入
- “十九八七”的设计目标（5%误差）
  - MIPS Linux应用：静态/动态翻译，效率100%
  - ARM Android应用：静态/动态翻译，效率90%
  - X86 Linux应用：动态翻译，效率80%
  - X86 Windows系统：动态翻译，效率70%

Loongarch vs. MIPS

- SPEC CPU2000 train运行时间
- FPGA 20MHz
- GCC-8.3.0
  - **Loongarch**

    -O3

    -mabi=lp64

    -march=loongarch
  - **MIPS**

    -O3

    -mabi=64

    -march=gs464v

Benchmark	Loongarch	MIPS	提升比率
164.gzip	1623.69	2054.27	26.5%
175.vpr	989.85	1231.82	24.4%
176.gcc	137.28	169.45	23.4%
181.mcf	1562.80	1529.20	-2.2%
186.crafty	665.22	762.06	14.6%
197.parser	368.64	401.59	8.9%
252.eon	278.75	296.95	6.5%
253.perlbmk	2011.25	2890.89	43.7%
254.gap	273.16	301.05	10.2%
255.vortex	372.09	465.18	25.0%
256.bzip2	1245.67	1333.68	7.1%
300.twolf	535.01	632.69	18.3%
SPECint 2000	619.76	722.87	16.6%
168.wupwise	1249.93	1258.34	0.7%
171.swim	422.59	449.71	6.4%
172.mgrid	487.16	502.92	3.2%
173.applu	495.58	518.90	4.7%
177.mesa	1226.82	1483.01	20.9%
178.galgel	529.07	593.07	12.1%
179.art	237.78	261.91	10.1%
183.equake	1194.57	1195.03	0.0%
187.facerec	1563.75	1591.55	1.8%
188.ammp	2066.05	2163.12	4.7%
189.lucas	1617.61	1942.31	20.1%
191.fma3d	5630.29	6007.80	6.7%
200.sixtrack	2208.14	2672.33	21.0%
301.apsi	372.18	410.95	10.4%
SPECfp 2000	1006.15	1100.90	9.4%

Linux进程级MIPS二进制翻译效果

- 初步结果，还没有完全稳定，还没有优化
  - FPGA 20MHz
  - SPEC CPU2000 train运行时间（秒）
  - 编译器为GCC 8.3.0 -O3
- 百分比为性能比例，越大越好，目标为100%

测试程序	MIPS	LATM	LA	LA/LATM	MIPS/LATM
SPEC CPU2000/164.gzip	2137	2082	1645	79%	103%
SPEC CPU2000/181.mcf	1518	1784	1599	90%	85%
SPEC CPU2000/197.parser	410	403	345	86%	102%
SPEC CPU2000/252.eon	308	329	259	79%	94%

Linux进程级X86二进制翻译效果

- SPEC CPU2000 ref运行时间
- 龙芯3A4000，主频1.8GHz
  - 在MIPS基础上增加X86二进制翻译支持
- 编译选项均为GCC -O2
  - 表中本地分值1885和1827不是最高值
  - Base本地分值为2071和2537分
- 优化目标为二进制翻译效率80%
  - 还有较大提升空间
  - QEMU浮点翻译效率很低，大量使用helper函数

程序名	本地	LATX	效率(%)	QEMU4.1	效率(%)	提升(%)
164.gzip	1010	605	59.9%	247	24.5%	244.9%
175.vpr	1509	643	42.6%	137	9.1%	469.3%
176.gcc	2260	722	31.9%	260	11.5%	277.7%
181.mcf	1675	2249	134.3%	965	57.6%	233.1%
186.crafty	2088	602	28.8%	249	11.9%	241.8%
197.parser	1576	763	48.4%	260	16.5%	293.5%
252.eon	2689	727	27.0%	41.7	1.6%	1743.4%
253.perlbmk	1672	697	41.7%	237	14.2%	294.1%
254.gap	1898	904	47.6%	269	14.2%	336.1%
255.vortex	3564	881	24.7%	301	8.4%	292.7%
256.bzip2	1524	747	49.0%	293	19.2%	254.9%
300.twolf	2283	1354	59.3%	435	19.1%	311.3%
定点平均	**1885**	**837**	**44.4%**	**249**	**13.2%**	**336.6%**
168.wupwise	2304	738	32.0%	23.3	1.0%	3167.4%
171.swim	678	1872	276.1%	43.8	6.5%	4274.0%
172.mgrid	1446	691	47.8%	11.7	0.8%	5906.0%
173.applu	1264	1065	84.3%	22.7	1.8%	4691.6%
177.mesa	2467	255	10.3%	55.3	2.2%	461.1%
178.galgel	3439	2299	66.9%	37.5	1.1%	6130.7%
179.art	5432	3488	64.2%	100	1.8%	3488.0%
183.equake	1739	1449	83.3%	39.8	2.3%	3640.7%
187.facerec	2667	1524	57.1%	54.9	2.1%	2776.0%
188.ammp	1644	903	54.9%	24.1	1.5%	3746.9%
189.lucas	1158	1312	113.3%	34.4	3.0%	3814.0%
191.fma3d	1876	1080	57.6%	30.5	1.6%	3541.0%
200.sixtrack	1049	391	37.3%	5.58	0.5%	7007.2%
301.apsi	2016	1090	54.1%	30.3	1.5%	3597.4%
浮点平均	**1827**	**1068**	**58.4%**	**31.3**	**1.7%**	**3411.3%**

Windows系统级X86二进制翻译效果

- 暂缺，目标效率为70%。
  - 最难的地址直接翻译（X86虚地址直接翻译成Loongarch物理地址）已经调通。

Loongarch知识产权的考虑

- 自主定义指令反而会凸显知识产权风险
  - 原来MIPS授权起到“保护伞”作用
  - 指令系统知识产权主要包括商标权、专利权（保护期限20年）、著作权（保护期限50年）
- 商标权
  - 已申请LoongArch商标，以后龙芯手册及软件均用LoongArch或简写LArch
- 专利权
  - 通用型指令2000年前已经成熟，最常用向量指令2000年前在X86的MMX、SSE2中已经出现
  - 专用向量指令（如AI指令）、虚拟机、多线程相关风险较大，进行重点规避
- 著作权
  - 主要涉及指令系统手册
  - 已采用不同风格重新编写指令系统手册

目前进展

- 已经完成龙芯GS132、GS264、GS464三大系列IP核指令系统修改
  - 基于Loongarch的某龙芯CPU已于2020Q2交付流片，2020Q4样片
- 基础软件OS
  - 完成BIOS、编译器、内核改造，在FPGA平台上运行SPEC CPU等复杂应用
  - 正在开展完整操作系统编译工作
  - 正在开展Java、JavaScript、.NET虚拟机的迁移工作
- 二进制翻译系统LAT基本完成开发，开始调试优化
  - MIPS和X86用户态二进制翻译持续改进中
  - X86系统态二进制翻译已经基本跑通，最难的地址翻译已经调通
- 争取2020年底前基本完成上述工作

未来计划

- 对Loongarch指令系统进行知识产权分析
  - 已委托权威第三方机构进行，预计2020年底完成国内部分，2021年完成国际部分
- 组建自主指令系统联盟
  - 龙芯将把Loongarch免费开放
  - 龙芯将把部分处理器IP核（Cortex A53以下性能）源码免费开放
  - 条件：联盟内企业互相不发生指令系统诉讼、形成对第三方的CPU防御联盟
  - 简化Loongarch，形成百条指令的小系统，在高校推广
- 持续改进二进制翻译的硬件支持和软件优化
  - 争取到2025年消除指令系统间的壁垒
- 2020年Q2起新流片的龙芯CPU均支持Loongarch，不再支持MIPS
  - 2020年前已流片产品继续维护到2030年

小结

- 建立自主信息产业体系需要自主指令系统
- 指令系统既自主又兼容是可以的，需要对计算机系统融会贯通（3+3+3）
  - 三个基础软件：BIOS、内核、编译器（GCC、LLVM、GOLANG）
  - 三个虚拟机：Java、JavaScript、.NET
  - 三个二进制翻译系统：MIPS、X86、ARM
- 龙芯经过多年积累，已经掌握上述3+3+3能力
  - “第一个百年”目标：支持Loongarch的龙芯下一代CPU和OS产品发布，龙芯平台原有应用可直接无损翻译运行，在龙芯Linux平台上高效运行Linux平台的X86应用如商用EDA工具，在龙芯Linux平台上运行Windows操作系统及相关应用
- 争取在2025年通过软硬融合的体系结构翻译消除指令系统壁垒
  - 需要持续的工程投入和迭代改进

将自主进行到底！

我们正在前进。我们正在做我们的前人从来没有做过的极其光荣伟大的事业（建立自主信息产业体系）。我们的目的一定要达到。我们的目的一定能够达到。