**二进制翻译教学系列**

二进制翻译是什么

二进制翻译软件，我们可以将它们看成模拟器。举个例子，在x86平台上将arm程序的机器码翻译成x86的，来实现在x86上运行arm程序的功能。正式的来说，二进制翻译指的是将计算机程序的二进制代码从一个平台或架构转换到另一个平台或架构的过程。这种转换使得原本为一种处理器或操作系统编写的程序能够在不同的处理器或操作系统上运行，而无需重新编译源代码。

二进制翻译可以用于多种目的，包括但不限于：

允许软件在不同的硬件架构上运行，使得为一个操作系统编译的程序能够在另一个操作系统上运行。

在某些情况下，二进制翻译可以优化程序的性能，特别是在目标平台上有特定优化时。

在某些安全敏感的环境中，二进制翻译可以用来隔离和保护执行环境，防止恶意代码的执行。

二进制翻译可以通过不同的技术实现，包括动态二进制翻译、静态二进制翻译等。动态二进制翻译在程序运行时进行，而静态二进制翻译在程序执行前完成。这些技术在虚拟机、模拟器和某些类型的中间件中非常常见。

二进制翻译的现状

目前成熟的二进制翻译软件有什么

他们各自的优缺点和适配范围

根据搜索结果，以下是一些成熟的二进制翻译软件及其各自的优缺点和适配范围：

1. \*\*Usercorn\*\*

- \*\*优点\*\*：

- 轻量级，不需要完整虚拟机，资源消耗低。

- 模块化，可扩展的插件系统，可以自由定制行为。

- 跨平台，基于Go语言，可在多种操作系统上运行。

- 灵活的二进制兼容性，支持多种处理器架构的二进制代码。

- 源码开放，社区活跃，易于贡献和定制。

- \*\*缺点\*\*：

- 由于是开源项目，可能需要一定的技术背景来使用和定制。

- \*\*适配范围\*\*：

- 适用于学术研究、漏洞挖掘、逆向工程以及软件测试等场景。

2. \*\*exagear\*\*

- \*\*优点\*\*：

- 纯软件就能达到很高的翻译效率。

- 经过多年的持续开发和优化，翻译运行x86程序的效率很惊艳。

- \*\*缺点\*\*：

- 需要破解才能在非华为CPU上运行。

- \*\*适配范围\*\*：

- 可以在ARM架构CPU上运行x86程序，经过破解后可以在任何ARM64架构CPU上运行。

3. \*\*龙芯二进制翻译技术\*\*

- \*\*优点\*\*：

- 技术升级后，系统占用率降低，安装包体积缩小，安装步骤简化。

- 支持大量外设和应用软件的适配。

- 提高了软件流畅性与稳定性。

- \*\*缺点\*\*：

- 相对于商业软件，可能在性能上还有提升空间。

- \*\*适配范围\*\*：

- 支持Linux到Linux，Linux/MIPS到Linux/LoongArch的翻译，以及Linux/X86到Linux/LoongArch的开发中。

- 支持Windows到Linux，包括Windows/X86应用的兼容运行。

4. \*\*QEMU\*\*

- \*\*优点\*\*：

- 支持广泛的硬件架构模拟和二进制翻译。

- 适用于研究和开发环境。

- \*\*缺点\*\*：

- 性能可能不如专门的二进制翻译工具。

- \*\*适配范围\*\*：

- 支持多种操作系统和硬件架构的模拟。

这些二进制翻译软件各有特点，选择时需要根据具体需求和环境来决定使用哪一个。

能否流畅的运行一些办公软件或者游戏

WPS

蔚蓝

我能不能实现一个二进制翻译软件

一个最简单的方法就是手动一个一个弄，比如指定 ADD EAX,EBX 翻译成ADD t0,t1。但是很明显这样做是不好的

尝试给出一些小型项目的学习地址（比如一些开源项目   
根据您的请求，以下是一些开源的二进制翻译软件项目，以及它们的学习地址：

1. \*\*Usercorn\*\*

- 这是一个轻量级的、可扩展的动态二进制翻译（Dynamic Binary Translation, DBT）引擎，用于在用户空间执行各种操作系统的内核代码和二进制程序。

- 项目地址：[Usercorn GitHub](https://gitcode.com/gh\_mirrors/us/usercorn)

2. \*\*Triton\*\*

- 一个动态二进制分析库，用于构建自己的程序分析工具，自动化逆向工程，执行软件验证或仅模拟代码。

- 项目地址：[Triton GitHub](https://github.com/topics/binary-translation)

3. \*\*Revng\*\*

- 一款静态二进制翻译工具，能够接受多种架构的ELF格式二进制文件，并将其转化为等价的LLVM IR中间表示。

- 项目地址：[Revng GitHub](https://gitcode.com/gh\_mirrors/re/revng)

这些项目都是开源的，您可以访问它们的GitHub页面来获取更多信息，包括文档、源代码和社区讨论，以便进一步学习和使用。

展示一下效果，对比成熟的翻译软件有什么差距

根据搜索结果，我们可以对比一些二进制翻译软件与成熟翻译软件之间的差距：

1. \*\*ExaGear\*\*

- \*\*优点\*\*：

- ExaGear能够将原本在不同架构上运行的程序转换为目标架构的指令，实现跨平台的运行。

- 优化后的AArch64本地程序能够达到原生性能的90%以上，性能损失相对较小。

- 支持广泛的操作系统和处理器架构，几乎可以覆盖所有主流平台。

- 提供良好的兼容性和稳定性，简单易用。

- \*\*缺点\*\*：

- 由于虚拟机的性质，可能会消耗一定的系统资源和内存，导致一些性能上的损失。

- 对于某些复杂的软件可能无法提供完全的兼容性，可能会出现部分功能无法正常运行的情况。

- \*\*与成熟软件的差距\*\*：

- ExaGear在性能上接近原生应用，但在资源消耗和兼容性方面可能不如一些成熟的虚拟机软件。

2. \*\*Triton\*\*

- \*\*优点\*\*：

- 高性能，独特的架构设计和优化策略使得它在处理大量数据和高并发场景时表现出色。

- 灵活性，提供丰富的API和扩展接口，支持多种编程语言和框架。

- 社区支持，拥有庞大的社区资源，遇到问题时可以获得及时的帮助和解决方案。

- \*\*缺点\*\*：

- 学习曲线较陡峭，新手可能需要花费一定的时间来熟悉和掌握。

- 对硬件资源的需求较高，如果项目预算有限或运行环境资源紧张，可能不是最佳选择。

- \*\*与成熟软件的差距\*\*：

- Triton在性能和灵活性方面具有优势，但可能在易用性和资源消耗方面与一些成熟的软件存在差距。

3. \*\*Usercorn 和 Revng\*\*

- 这两个项目主要是研究性质的，可能在实际应用和成熟度方面与商业软件有较大差距。它们可能更适合学术研究、漏洞挖掘、逆向工程以及软件测试等场景，而不是作为成熟的商业解决方案。

综上所述，这些二进制翻译软件在特定领域和应用场景下具有优势，但在易用性、资源消耗和兼容性方面可能与成熟的商业软件存在差距。选择合适的二进制翻译软件应根据项目的具体需求、团队的规模和预算等因素综合考虑。

结合 Box64 如何去实现

一条 RISCV 指令在 Box64 上怎么翻译去执行

在Box64上，将RISCV指令翻译去执行的过程主要涉及以下几个方面：

1. \*\*动态重编译（DynaRec）\*\*：

Box64的JIT引擎DynaRec负责将x86\_64指令翻译成RISC-V指令。对于每个需要执行的x86\_64指令，DynaRec会生成一系列RISC-V指令来实现相同的逻辑。这个过程涉及到将x86\_64的opcodes映射到RISC-V的opcodes上，实现1:1的指令转换。

2. \*\*寄存器映射\*\*：

所有的x86\_64寄存器都被映射到特定的RISC-V寄存器上，这种1:1的映射方式简化了翻译过程。

3. \*\*位操作和内存操作\*\*：

RISC-V在处理位操作和内存操作时与x86\_64有所不同。例如，对于`ADD AH, BL`这样的指令，Box64需要从RBX中提取最低字节，加到RAX的第二低字节上，然后再将结果插回到RAX的第二低字节，同时保持RAX中其他字节不变。在RISC-V上，这需要通过多个指令来实现，因为RISC-V没有直接的位字段提取和插入指令，需要通过一系列的逻辑操作来完成。

4. \*\*原子操作\*\*：

x86有LOCK前缀的指令用于锁自由的原子操作，Box64主要使用LR/SC序列来模拟这些操作。对于16字节的原子指令，如`LOCK CMPXCHG16B`，RISC-V没有直接的对应指令，因此Box64无法像其他架构那样完美地实现这些指令。

5. \*\*性能优化\*\*：

Box64通过目标架构的动态链接库重新实现了部分x86\_64动态链接库，以避免二进制翻译的性能损失，并降低对x86\_64环境的依赖。同时，需要在中间对传入的参数和返回值进行翻译，以适应x86\_64和RV64间的调用约定差异。

6. \*\*SIMD支持\*\*：

Box64对RISC-V的SIMD支持正在逐步增强。最近，PLCT Lab为Box64的RISC-V后端添加了对RVV 1.0的初步支持，使得Box64能够将超过100个SSE指令高效地翻译成RVV指令。这显著提高了性能，特别是在运行大量使用SSE指令的x86\_64程序时。

综上所述，Box64通过DynaRec动态重编译技术，将x86\_64指令翻译成RISC-V指令执行，同时在位操作、内存操作、原子操作和SIMD支持方面进行了特定的优化和适配，以实现在RISC-V架构上高效运行x86\_64程序。

为了进一步提高性能，Box64通过目标架构的动态链接库重新实现了部分x86\_64动态链接库，因此这部分的函数调用会进入目标架构原生的动态链接库中，避免了二进制翻译的性能损失，同时也降低了对x86\_64环境的依赖。但是x86\_64和目标架构的调用约定存在差异，需要在中间对传入的参数和返回值进行翻译。实现DynaRec RV64的后端的同时也需要完善x86\_64和RV64间的调用约定转换。

目前的 RISCV 社区对二进制翻译软件的支持

根据搜索结果，RISC-V社区对二进制翻译软件的支持主要体现在以下几个方面：

1. \*\*openKylin社区的积极参与\*\*：

- openKylin社区的RISC-V SIG团队积极参与二进制翻译工作，并参与开源项目box64的研发。他们已经提交并合并了20多个PR（代码审查请求），增加了GTK3、nettle等多个动态库及函数的封装，以及对RISC-V动态重编译模块等功能的完善，总计贡献了2000余行代码。

2. \*\*成功案例\*\*：

- RISC-V SIG团队通过解决指令翻译、函数库封装等关键问题，成功在算能SG 2042平台和openKylin 2.0 RISC-V版本系统中运行了X86架构软件，包括微信、WPS Office和腾讯会议等。

3. \*\*性能和兼容性的持续改进\*\*：

- 尽管openKylin在RISC-V二进制翻译方面取得了一定成果，但在性能、兼容性等方面仍存在不足。未来，RISC-V SIG将持续进行二进制翻译的技术探索，并积极参与并贡献于box64等开源项目的建设。

4. \*\*软件生态的构建\*\*：

- openKylin社区已经适配完成国内外主流RISC-V硬件平台，并构建了较为完整的软件生态体系。他们率先完成了钉钉、搜狗输入法和福昕等商用软件的移植，并针对AI应用与开发进行了优化。

5. \*\*开放二进制翻译联盟的成立\*\*：

- 在RISC-V中国峰会上，宣布了开放二进制翻译联盟的成立，力求打通RISC-V和诸多指令集架构之间的开放访问，这对于二进制翻译软件的支持是一个重要的里程碑。

6. \*\*社区的号召和参与\*\*：

- RISC-V SIG主要负责社区RISC-V架构开源软件包的维护，发布openKylin的RISC-V版本，进行软件包构建、系统构建等工作，并欢迎对RISC-V开发平台技术方向感兴趣的爱好者加入。

综上所述，RISC-V社区对二进制翻译软件的支持表现在积极的技术探索、成功的案例实现、性能和兼容性的持续改进、软件生态的构建以及开放二进制翻译联盟的成立等多个方面。这些努力共同推动了RISC-V架构在软件兼容性和生态建设方面的发展。