ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences
智能软件研究中心 Intelligent Software Research Center

QEMU TCG Plugins 介绍和使用

程序语言与编译技术实验室(PLCT) 王俊强

目录

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

QEMU

QEMU TCG Plugins

TCG Plugins是自4.2以来的一个新特性，它提供了在代码上运行测试的能力。它们能够对系统发出的每个指令和内存访问进行被动监视。

NOT TPI(TCG Plugin Interface)

来源:https://wiki.qemu.org/Features/TCGPlugins

TCG Plugins源码

功能源码	接口API	测试Case
accel/tcg/Makefile.objs accel/tcg/atomic_common.inc.c accel/tcg/atomic_template.h accel/tcg/cpu-exec.c accel/tcg/cputlb.c accel/tcg/plugin-gen.c accel/tcg/plugin-helpers.h accel/tcg/translate-all.c accel/tcg/translator.c accel/tcg/user-exec.c ……	plugins/Makefile.objs plugins/api.c plugins/core.c plugins/loader.c plugins/plugin.h plugins/qemu-plugins.symbols ……	tests/plugin/Makefile tests/plugin/bb.c tests/plugin/empty.c tests/plugin/hotblocks.c tests/plugin/hotpages.c tests/plugin/howvec.c tests/plugin/insn.c tests/plugin/mem.c ……

头文件：include/exec
include/qemu

参考： https://www.mail-archive.com/qemu-devel@nongnu.org/msg656472.html

Building and Runing

- 编译方法

  功能开启: --enable-plugins

  plugins编译: configure --enable-plugins => make –C tests/plugins

  add file: tests/plugins/demo.c
  Makefile: NAMES += demo

- 运行方法

```
$QEMU $OTHER_QEMU_ARGS \
        -plugin tests/plugin/libhowvec.so,arg=inline,arg=hint \
        -plugin tests/plugin/libhotblocks.so    -D output   -d plugin
```

作用对象

- Translation block(TB块)

  操作 data types: qemu_plugin_tb

- Instruction

  操作 data types: qemu_plugin_insn

- Memory

  操作 data types: qemu_plugin_hwaddr

  qemu_plugin_meminfo_t

TCG plugins 主要结构体

```
struct qemu_plugin_tb {
    GPtrArray *insns;
    size_t n;
    uint64_t vaddr;
    uint64_t vaddr2;
    void *haddr1;
    void *haddr2;
    GArray *cbs[PLUGIN_N_CB_SUBTYPES];
};
```

insns: 指令集
vaddr: TB虚地址
cbs: 回调函数

```
struct qemu_plugin_insn {
    GByteArray *data;
    uint64_t vaddr;
    void *haddr;
    GArray *cbs[PLUGIN_N_CB_TYPES][PLUGIN_N_CB_SUBTYPES];
    bool calls_helpers;
    bool mem_helper;
};
```

data: 指令数据
vaddr：虚地址
haddr：物理地址
cbs: 回调函数

动态回调函数类型：

```
enum plugin_dyn_cb_type {
    PLUGIN_CB_INSN,
    PLUGIN_CB_MEM,
    PLUGIN_N_CB_TYPES,
};
```

```
enum plugin_dyn_cb_subtype {
    PLUGIN_CB_REGULAR,
    PLUGIN_CB_INLINE,
    PLUGIN_N_CB_SUBTYPES,
};
```

TCG plugins 主要结构体

```
typedef uint32_t qemu_plugin_meminfo_t;
struct qemu_plugin_hwaddr {
    bool is_io;
    bool is_store;
    union {
        struct {
            MemoryRegionSection *section;
            hwaddr    offset;
        } io;
        struct {
            uint64_t hostaddr;
        } ram;
    } v;
};
```

```
#define TRACE_MEM_SZ_SHIFT_MASK 0xf
/* size shift mask */
#define TRACE_MEM_SE (1ULL << 4)
/* sign extended (y/n) */
#define TRACE_MEM_BE (1ULL << 5)
/* big endian (y/n) */
#define TRACE_MEM_ST (1ULL << 6)
/* store (y/n) */
#define TRACE_MEM_MMU_SHIFT 8
/* mmu idx */
```

is_io: 是否在内存/缓存(TLB命中)
is_store: 是否存储
v: 内存/Host RAM地址

TCG plugins 主要结构体

```
struct qemu_plugin_ctx {
    GModule *handle;
    qemu_plugin_id_t id;
    struct qemu_plugin_cb *callbacks[QEMU_PLU
GIN_EV_MAX];
    QTAILQ_ENTRY(qemu_plugin_ctx) entry;
    /*
     * keep a reference to @desc until uninst
all, so that plugins do not have
     * to strdup plugin args.
     */
    struct qemu_plugin_desc *desc;
    bool installing;
    bool uninstalling;
    bool resetting;
};
```

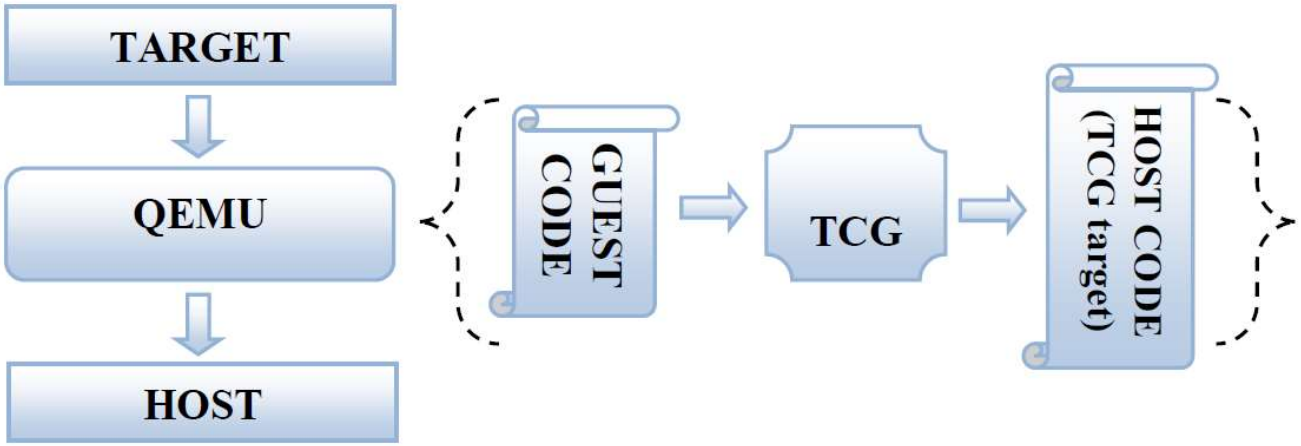
```
struct qemu_plugin_dyn_cb {
    union qemu_plugin_cb_sig f;
    void *userp;
    unsigned tcg_flags;
    enum plugin_dyn_cb_subtype type;
    /* @rw applies to mem callbacks only (bot
h regular and inline) */
    enum qemu_plugin_mem_rw rw;
    /* fields specific to each dyn_cb type go
 here */
    union {
        struct {
            enum qemu_plugin_op op;
            uint64_t imm;
        } inline_insn;
    };
};
```

实现依赖于:QEMU HELPER？？？

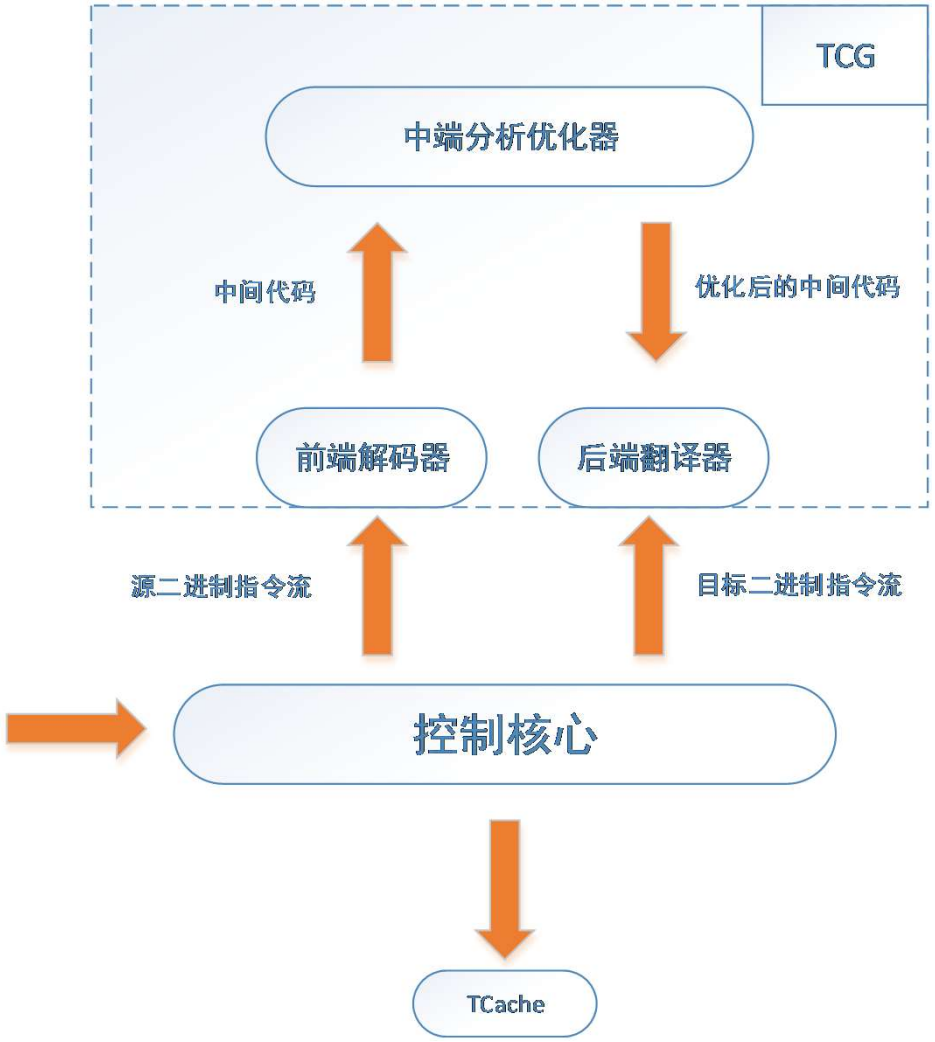
QEMU TCG

微代码生成器（TCG）

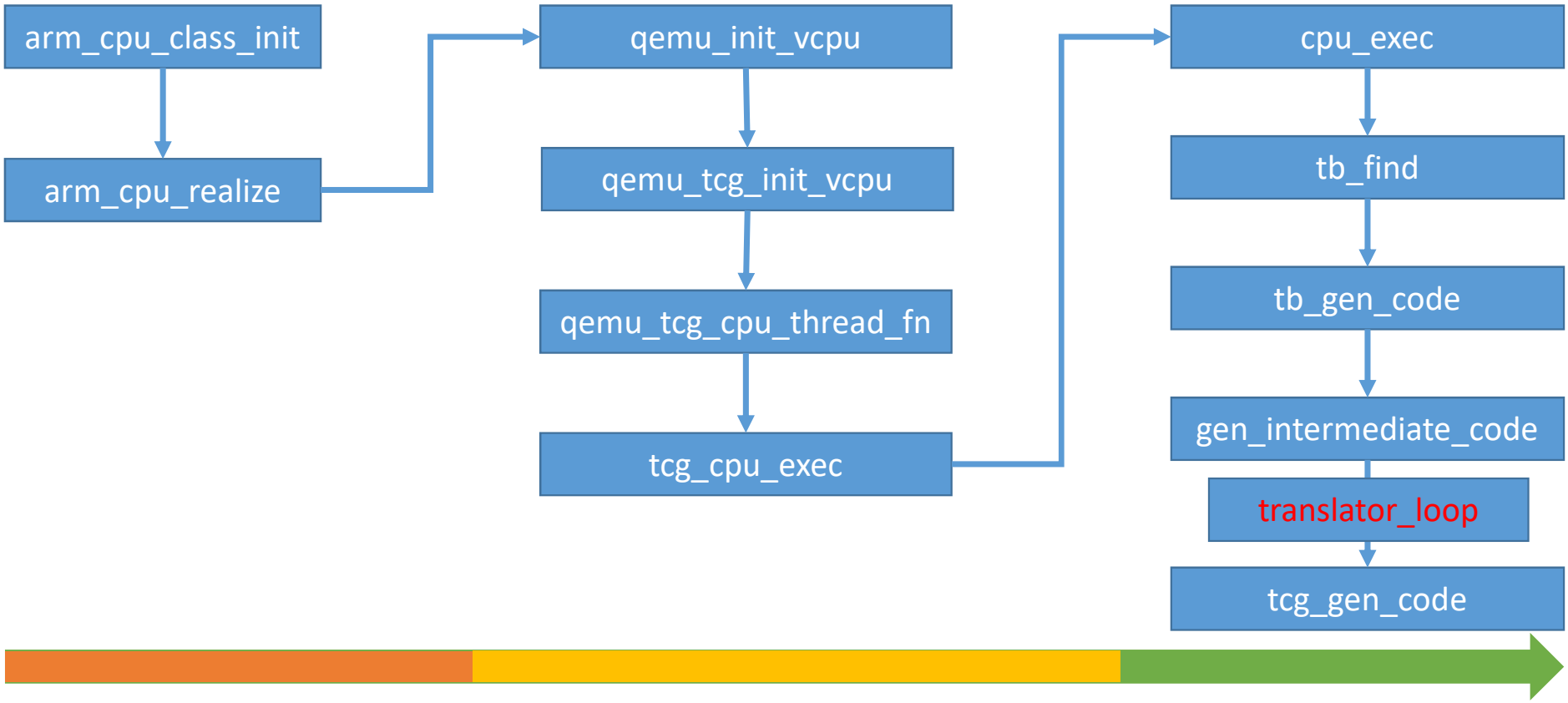
Tiny Code Generator（TCG）将源处理器机器代码转换为虚拟机运行所需的机器代码块(如x86机器代码块)



源自：Qemu Detailed Study


TCG
中端分析优化器
中间代码
优化后的中间代码
前端解码器
后端翻译器
源二进制指令流
目标二进制指令流
控制核心
TCache

TCG转换流程



指定CPU设备初始化

tcg翻译和执行的线程

TB块的索引翻译

TCG plugins 主要接口

接口名称	类别	作用对象
qemu_plugin_register_vcpu_init_cb	Plugin Register Functions	vCPU
qemu_plugin_register_vcpu_exit_cb		vCPU
qemu_plugin_register_vcpu_idle_cb		vCPU
qemu_plugin_register_vcpu_resume_cb		vCPU
qemu_plugin_register_vcpu_tb_exec_cb		Translation block
qemu_plugin_register_vcpu_tb_exec_inline		Translation block
qemu_plugin_register_vcpu_insn_exec_cb		Instruction
qemu_plugin_register_vcpu_insn_exec_inline		Instruction
qemu_plugin_register_vcpu_mem_cb		Memory Instrumentation
qemu_plugin_register_vcpu_mem_inline		Memory Instrumentation
qemu_plugin_register_vcpu_tb_trans_cb		Translation block
qemu_plugin_register_vcpu_syscall_cb		instruction
qemu_plugin_register_vcpu_syscall_ret_cb		instruction

TCG plugins 主要接口

插件注册事件 9个：

```
enum qemu_plugin_event {
    QEMU_PLUGIN_EV_VCPU_INIT,
    QEMU_PLUGIN_EV_VCPU_EXIT,
    QEMU_PLUGIN_EV_VCPU_TB_TRANS,
    QEMU_PLUGIN_EV_VCPU_IDLE,
    QEMU_PLUGIN_EV_VCPU_RESUME,
    QEMU_PLUGIN_EV_VCPU_SYSCALL,
    QEMU_PLUGIN_EV_VCPU_SYSCALL_RET,
    QEMU_PLUGIN_EV_FLUSH,
    QEMU_PLUGIN_EV_ATEXIT,
    QEMU_PLUGIN_EV_MAX,
};
```

回调函数CPU REGS权限：

```
enum qemu_plugin_cb_flags {
    QEMU_PLUGIN_CB_NO_REGS,
    QEMU_PLUGIN_CB_R_REGS,
    QEMU_PLUGIN_CB_RW_REGS???,
};
```

inline op：

```
enum qemu_plugin_op {
    QEMU_PLUGIN_INLINE_ADD_U64,
};
```

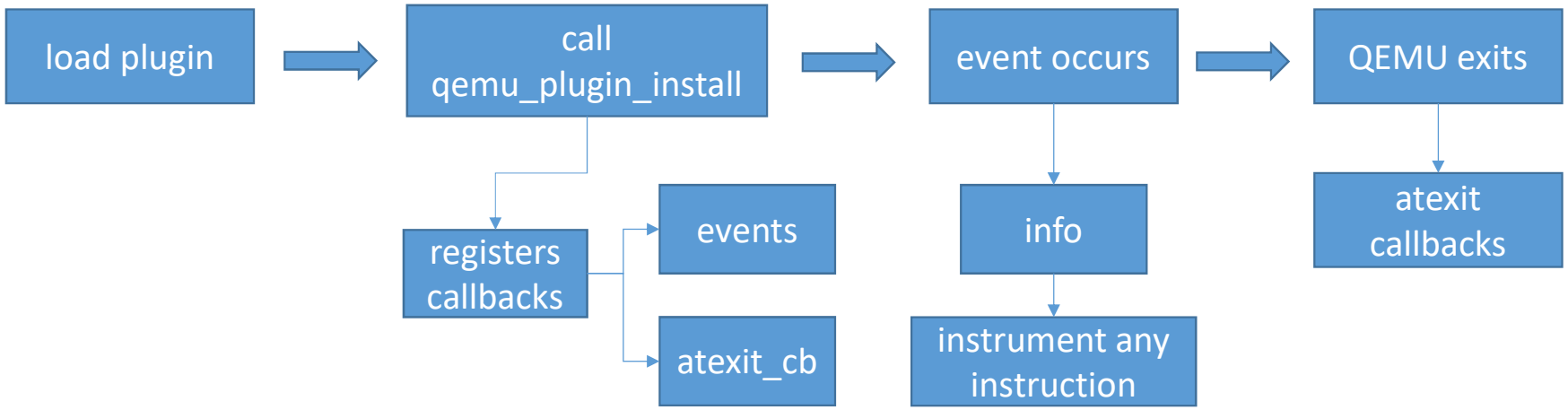
TCG plugins 主要接口

接口名称	类别	作用对象
qemu_plugin_tb_n_insns	Plugin Queries	Translation block
qemu_plugin_tb_vaddr		Translation block
qemu_plugin_tb_get_insn		Translation block
qemu_plugin_insn_data		Instruction information
qemu_plugin_insn_size		Instruction information
qemu_plugin_insn_vaddr		Instruction information
qemu_plugin_insn_haddr		Instruction information
qemu_plugin_insn_disas		Instruction information
qemu_plugin_mem_size_shift		memory access
qemu_plugin_mem_is_sign_extended		memory access
qemu_plugin_mem_is_big_endian		memory access
qemu_plugin_mem_is_store		memory access

TCG plugins 主要接口

接口名称	类别	作用对象
qemu_plugin_get_hwaddr	Plugin Queries	Virtual Memory
qemu_plugin_hwaddr_is_io		Virtual Memory
qemu_plugin_hwaddr_to_raddr？？？		Virtual Memory
qemu_plugin_hwaddr_device_offset		Virtual Memory
qemu_plugin_vcpu_for_each		vCPUs
qemu_plugin_n_vcpus		vCPUs
qemu_plugin_n_max_vcpus		vCPUs
qemu_plugin_outs	Plugin Info output	Plugin
qemu_plugin_uninstall	Uninstall handlers	Plugin
qemu_plugin_reset	Reset handlers	Plugin
qemu_plugin_register_atexit_cb	Exit handlers	Plugin
qemu_plugin_install	Start handlers	Plugin

TCG plugins 主要流程



TCG plugins 主框架

源码：tests/plugin/empty.c

```c
#include <qemu-plugin.h>

QEMU_PLUGIN_EXPORT int qemu_plugin_version = QEMU_PLUGIN_VERSION;

static void vcpu_tb_trans(qemu_plugin_id_t id, struct qemu_plugin_tb *tb)
{ }

QEMU_PLUGIN_EXPORT int qemu_plugin_install(qemu_plugin_id_t id,
                                           const qemu_info_t *info,
                                           int argc, char **argv)
{
    qemu_plugin_register_vcpu_tb_trans_cb(id, vcpu_tb_trans);
    return 0;
}
```

TCG plugins示例

第一步:
Install函数

第二步:
实现功能回调函数

第三步:
实现退出回调

第四步:
实现Insn记录函数

第五步:
编译运行

源码：tests/plugin/insn.c

```
QEMU_PLUGIN_EXPORT int qemu_plugin_install(qemu_p
lugin_id_t id,const qemu_info_t *info,int argc, c
har **argv)
{
    if (argc && !strcmp(argv[0], "inline")) {
        do_inline = true;
    }

    qemu_plugin_register_vcpu_tb_trans_cb(id, vcp
u_tb_trans);
    qemu_plugin_register_atexit_cb(id, plugin_exi
t, NULL);
    return 0;
}
```

TCG plugins示例

第一步:
Install函数

第二步:
实现功能回调

第三步:
实现退出回调

第四步:
实现Insn记录函数

第五步:
编译运行

```
static void vcpu_tb_trans(qemu_plugin_id_t id, struct qemu_plug
in_tb *tb)
{
    size_t n = qemu_plugin_tb_n_insns(tb);
    size_t i;

    for (i = 0; i < n; i++) {
        struct qemu_plugin_insn *insn = qemu_plugin_tb_get_insn
(tb, i);
        char *insn_disas = qemu_plugin_insn_disas(insn);
        printf("insn: %s\n", insn_disas);

        if (do_inline) {
            qemu_plugin_register_vcpu_insn_exec_inline(
                insn, QEMU_PLUGIN_INLINE_ADD_U64, &insn_count,
1);
        } else {
            qemu_plugin_register_vcpu_insn_exec_cb(
                insn, vcpu_insn_exec_before, QEMU_PLUGIN_CB_NO_
REGS, NULL);
        }
    }
}
```

TCG plugins示例

第一步:
Install函数

第二步:
实现功能回调函数

第三步:
实现退出回调

第四步:
实现Insn记录函数

第五步:
编译运行

```
static void plugin_exit(qemu_plugin_id_t id, void
 *p)
{
    g_autofree gchar *out;
    out = g_strdup_printf("insns: %" PRIu64 "\n",
 insn_count);
    qemu_plugin_outs(out);
}
```

TCG plugins示例

第一步:
Install函数

第二步:
实现功能回调函数

第三步:
实现退出回调

第四步:
实现Insn记录函数

第五步:
编译运行

```
static uint64_t insn_count;

static void vcpu_insn_exec_before(unsigned int cp
u_index, void *udata)
{
    insn_count++;
}
```

TCG plugins示例

第一步:
Install函数

第二步:
实现功能回调函数

第三步:
实现退出回调

第四步:
实现Insn记录函数

第五步:
编译运行

测试APP Case: tests/tcg/arm-linux-user/hello-arm

arm-linux-user/qemu-arm -d plugin -plugin tests/plugin/libinsn.so,arg=inline tests/tcg/arm-linux-user/hello-arm

```
wang@lelouch:~/WorkRoom/qemu/plct-qemu/build$ ./arm-linux-user/qemu-arm -d plugin -plugin t
ests/plugin/libinsn.so,arg=inline tests/tcg/arm-linux-user/hello-arm
insn: push {fp, lr}
insn: add fp, sp, #4
insn: mov r0, #1
insn: ldr r1, [pc, #0x10]
insn: mov r2, #0xc
insn: bl #4294967200
insn: push {r4, fp, lr}
insn: add fp, sp, #8
insn: sub sp, sp, #0x1c
insn: str r0, [fp, #-0x18]
insn: str r1, [fp, #-0x1c]
insn: str r2, [fp, #-0x20]
insn: ldr ip, [fp, #-0x1c]
insn: ldr r3, [fp, #-0x18]
insn: ldr r4, [fp, #-0x20]
insn: mov r0, r3
insn: mov r1, ip
insn: mov r2, r4
insn: svc #0x900004
Hello World
insn: mov r3, r0
insn: str r3, [fp, #-0x10]
insn: ldr r3, [fp, #-0x10]
insn: mov r0, r3
insn: sub sp, fp, #8
insn: pop {r4, fp, pc}
insn: mov r0, #0
insn: bl #4294967140
insn: push {fp, lr}
insn: add fp, sp, #4
insn: sub sp, sp, #0x10
insn: str r0, [fp, #-0x10]
insn: ldr r3, [fp, #-0x10]
insn: mov r0, r3
insn: svc #0x900001
insns: 34
```

谢谢

wangjunqiang@iscas.ac.cn