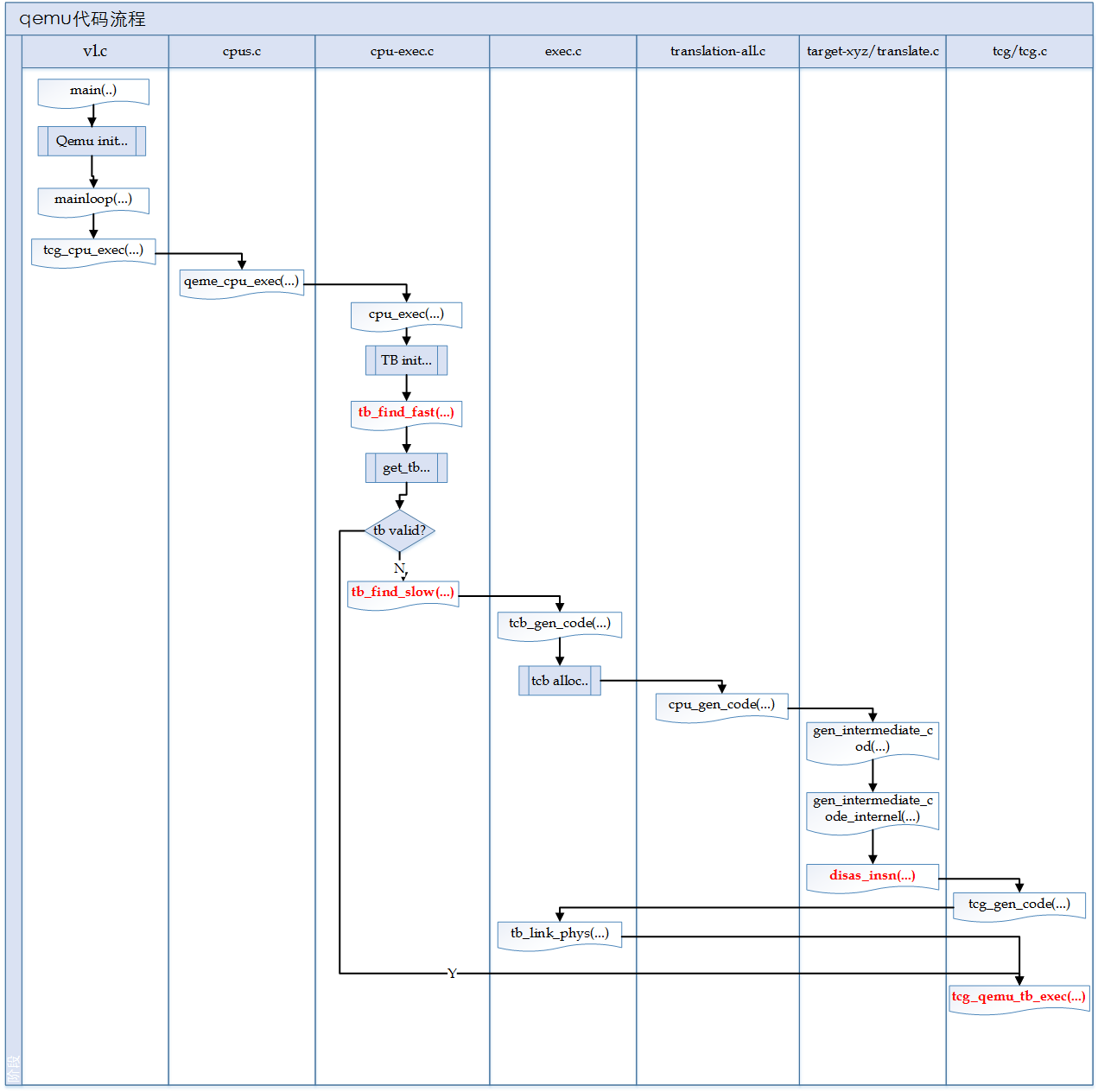
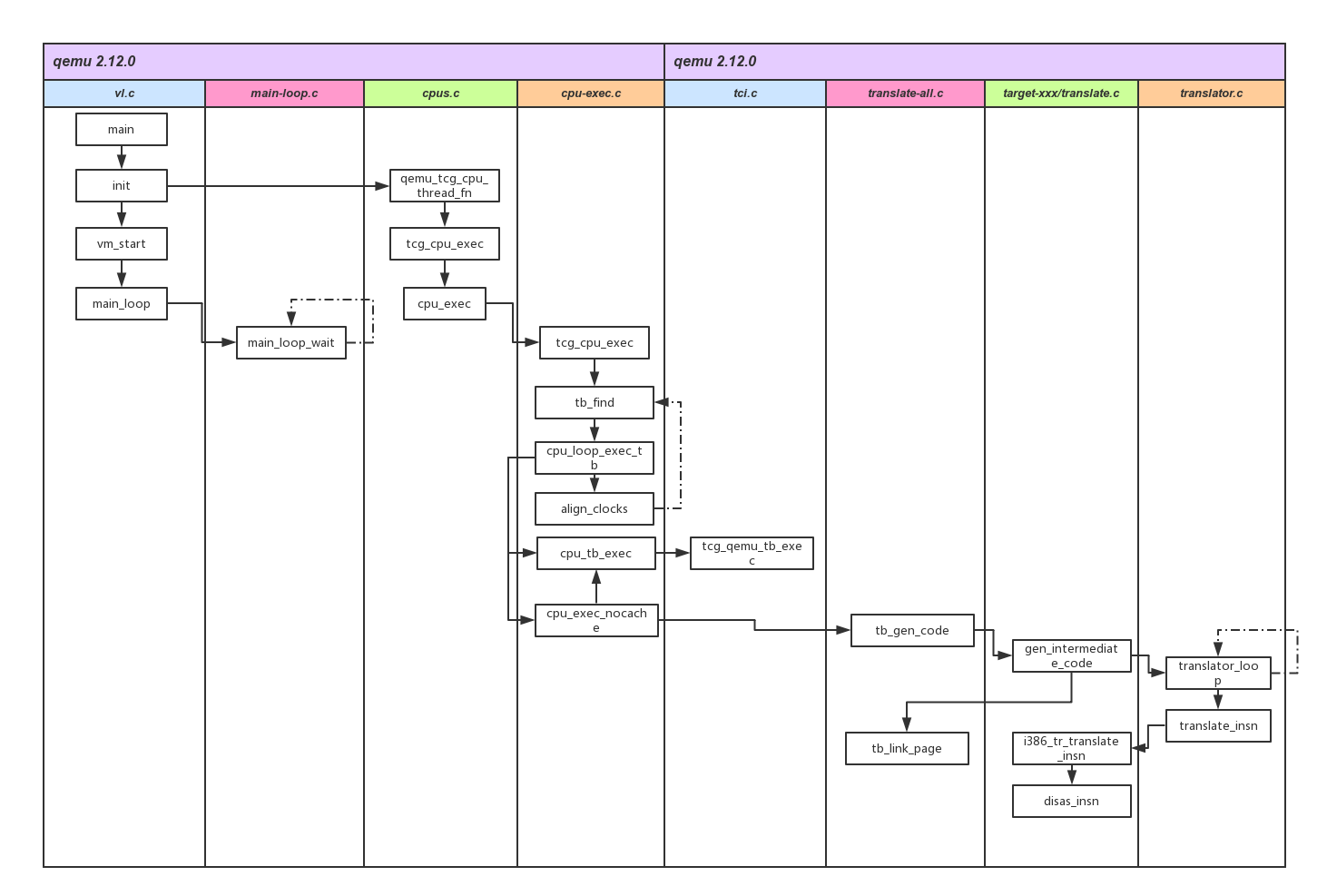
# qemu代码流程

下图是网上找到的，印象中之前的流程是这样的，但是在查看qemu 2.12.0源码时发现现在的qemu架构已经有些不同。



# qemu2.12.0的架构

在看流程之前首先要了解qemu的整体架构，QEMU实际上使用的是将事件驱动机制和线程相结合的一种混合架构，为每一个vCPU分配一个QEMU线程，以及一个专用的事件处理循环线程，这个模型称为iothread。详见https://www.ibm.com/developerworks/community/blogs/8b3f8d48-b83c-4bcb-a34e-2b9c8c13ab1f/entry/20161206?lang=en

# Pyrebox的架构

结合上面的qemu源码分析，可以找到合适的位置插装代码、提取信息。具体来讲，

## qemu/vl.c：

添加入了头文件pyrebox/pyrebox.h，解析pyrebox的设置，初始化pyrebox模块，主要是设置python环境。

## qemu/accel/tcg/cpu-exec.c：

添加了头文件pyrebox/qemu\_glue\_callbacks.h，更新记录的cpu执行状态，notify\_cpu\_executing()

## qemu/target/i386/translate.c：

主要功能的实现，大量改动，添加了头文件pyrebox/qemu\_glue\_callbacks\_needed.h，

1. 在DisasContext结构中添加了4个变量，

> target\_ulong pgd;

//Pyrebox: Save the PGD in the DisasContext to allow conditional instrumentation based on the PGD

target\_ulong pgd;

//Pyrebox: Save the pc between each pair of instructions, because when the insn\_end / block\_end is triggered, the enviroment CPU is already pointing to the next block

target\_ulong saved\_pc;

//Pyrebox: Save the opcode while doing the dissasembly, in order to call the corresponding opcode range callback.

uint32\_t saved\_opcode;

//Pyrebox: the cpu.

CPUState\* cs;

1. 在gen\_goto\_tb()、do\_gen\_eob\_worker()等函数，包括TB\_begin、insn\_begin中插装代码实现分析

//Pyrebox: insn end

gen\_helper\_qemu\_insn\_end\_callback()

//Pyrebox: block\_end

gen\_helper\_qemu\_block\_end\_callback()

//Pyrebox: opcode range

gen\_helper\_qemu\_opcode\_range\_callback()

1. 在gen\_illegal\_opcode()中插装代码实现qemu运行暂停
2. 在disas\_insn()中插装代码实现对指令的分析，特别是int3、intN的处理

## /target/i386/helper.h

添加了qemu help函数

//void qemu\_block\_begin\_callback(CPUState\* cpu,TranslationBlock\* tb);

DEF\_HELPER\_2(qemu\_block\_begin\_callback, void, ptr, ptr)

//void qemu\_op\_block\_begin\_callback(CPUState\* cpu,TranslationBlock\* tb);

//DEF\_HELPER\_2(qemu\_op\_block\_begin\_callback, void, ptr, ptr)

//void qemu\_op\_insn\_begin\_callback(CPUState\* cpu);

//DEF\_HELPER\_1(qemu\_op\_insn\_begin\_callback, void, ptr)

//void qemu\_block\_end\_callback(CPUState\* cpu,TranslationBlock\* next\_tb, target\_ulong from);

DEF\_HELPER\_4(qemu\_block\_end\_callback, void, ptr,ptr,tl,tl)

//void qemu\_insn\_begin\_callback(CPUState\* cpu);

DEF\_HELPER\_1(qemu\_insn\_begin\_callback, void, ptr)

//void qemu\_insn\_end\_callback(CPUState\* cpu);

DEF\_HELPER\_1(qemu\_insn\_end\_callback, void, ptr)

//void qemu\_opcode\_range\_callback(CPUState\* cpu, target\_ulong from, target\_ulong to, uint32\_t opcode);

DEF\_HELPER\_5(qemu\_opcode\_range\_callback, void, ptr,tl,tl,i32,tl)

DEF\_HELPER\_1(qemu\_trigger\_cpu\_loop\_exit\_if\_needed, void, ptr)

# python接口实现

另外，为了支持ipython，pyrebox做了对qemu做了额外的修改：

qemu/chardev/char-fd.c：添加了头文件pyrebox/pyrebox.h和qemu/main-loop.h，在这里处理了ipython和qemu原有的控制字符之间冲突的问题，利用pyrebox\_mutex实现