Ftrace Kernel Hooks: More than just tracing

Presenter:
Steven Rostedt
rostedt@goodmis.org
Red Hat

Ftrace Function Hooks

- Function Tracer
- Function Graph Tracer
- Function Profiler
- Stack Tracer
- Kprobes
- Uprobes
- Perf
- Pstore
- SystemTap

Function Tracing

```
# cd /sys/kernel/debug/tracing
# echo function > current_tracer
# cat trace
# tracer: function
#
# entries-in-buffer/entries-written: 205022/119956607   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <idle>-0     [002] dN.1  1781.978299: rcu_eqs_exit <-rcu_idle_exit
          <idle>-0     [002] dN.1  1781.978300: rcu_eqs_exit_common <-rcu_eqs_exit
          <idle>-0     [002] .N.1  1781.978301: arch_cpu_idle_exit <-cpu_startup_entry
          <idle>-0     [002] .N.1  1781.978301: tick_nohz_idle_exit <-cpu_startup_entry
          <idle>-0     [002] dN.1  1781.978301: ktime_get <-tick_nohz_idle_exit
          <idle>-0     [002] dN.1  1781.978302: update_ts_time_stats <-tick_nohz_idle_exit
          <idle>-0     [002] dN.1  1781.978302: nr_iowait_cpu <-update_ts_time_stats
          <idle>-0     [002] dN.1  1781.978303: tick_do_update_jiffies64 <-tick_nohz_idle_exit
          <idle>-0     [002] dN.1  1781.978303: update_cpu_load_nohz <-tick_nohz_idle_exit
          <idle>-0     [002] dN.1  1781.978303: calc_load_exit_idle <-tick_nohz_idle_exit
```

Function Graph Tracer

```
# echo function_graph > current_tracer
# cat trace
# tracer: function_graph
#
# CPU  DURATION                  FUNCTION CALLS
# |     |   |                     |   |   |   |
 2)   7.879 us    |  } /* context_tracking_user_exit */
 2)               |  __do_page_fault() {
 2)   0.070 us    |    down_read_trylock();
 2)   0.057 us    |    __might_sleep();
 2)   0.096 us    |    find_vma();
 2)               |    handle_mm_fault() {
 2)               |      __do_fault() {
 2)               |        filemap_fault() {
 2)               |          find_get_page() {
 2)   0.057 us    |            __rcu_read_lock();
 2)   0.061 us    |            __rcu_read_unlock();
 2)   1.241 us    |          }
 2)   0.074 us    |          __might_sleep();
 2)   2.201 us    |        }
 2)               |        _raw_spin_lock() {
 2)   0.069 us    |          preempt_count_add();
 2)   0.528 us    |        }
 2)   0.063 us    |        add_mm_counter_fast();
 2)   0.070 us    |        page_add_file_rmap();
 2)               |        _raw_spin_unlock() {
 2)   0.070 us    |          preempt_count_sub();
```

Dynamic Function Tracing

```
# echo '*sched*' > set_ftrace_filter
# echo function > current_tracer
# cat trace
# tracer: function
#
# entries-in-buffer/entries-written: 193727/240417   #P:4
#
#                              _-----=> irqs-off
#                             / _----=> need-resched
#                            | / _---=> hardirq/softirq
#                            || / _--=> preempt-depth
#                            ||| /     delay
#           TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#              | |       |   ||||       |         |
          <idle>-0     [003] d.h3  6325.742705: resched_task <-check_preempt_curr
          <idle>-0     [003] dNh3  6325.742712: native_smp_send_reschedule <-enqueue_task_fair
          <idle>-0     [003] dNh3  6325.742714: resched_task <-check_preempt_curr
          <idle>-0     [003] dN.1  6325.742719: smp_reschedule_interrupt <-reschedule_interrupt
          <idle>-0     [003] dN.1  6325.742720: scheduler_ipi <-smp_reschedule_interrupt
          <idle>-0     [003] dNh1  6325.742722: sched_ttwu_pending <-scheduler_ipi
          <idle>-0     [003] .N.1  6325.742728: schedule_preempt_disabled <-cpu_startup_entry
          <idle>-0     [003] .N..  6325.742729: schedule <-schedule_preempt_disabled
          <idle>-0     [003] .N..  6325.742731: __schedule <-preempt_schedule
          <idle>-0     [003] .N.1  6325.742732: rcu_sched_qs <-rcu_note_context_switch
          <idle>-0     [003] dN.2  6325.742733: pre_schedule_idle <-__schedule
           aprsd-3467  [003] ....  6325.742746: schedule <-do_nanosleep
           aprsd-3467  [003] ....  6325.742747: __schedule <-schedule
           aprsd-3467  [003] ...1  6325.742748: rcu_sched_qs <-rcu_note_context_switch
           aprsd-3454  [003] ....  6325.742767: schedule <-do_nanosleep
           aprsd-3454  [003] ....  6325.742767: __schedule <-schedule
           aprsd-3454  [003] ...1  6325.742768: rcu_sched_qs <-rcu_note_context_switch
     rcu_preempt-9     [003] d..2  6325.742788: smp_reschedule_interrupt <-reschedule_interrupt
     rcu_preempt-9     [003] d..2  6325.742789: scheduler_ipi <-smp_reschedule_interrupt
```

How it works?

- gcc’s profiler option: -pg
- Adds special mcount function call
    - all functions call mcount
    - mcount is a trampoline 蹦床

参考：https://rtoax.blog.csdn.net/article/details/116718210

A function call

```
asmlinkage __visible void __sched schedule(void)
{
    struct task_struct *tsk = current;

    sched_submit_work(tsk);
    __schedule();
}
```

A function call

- Disassembled

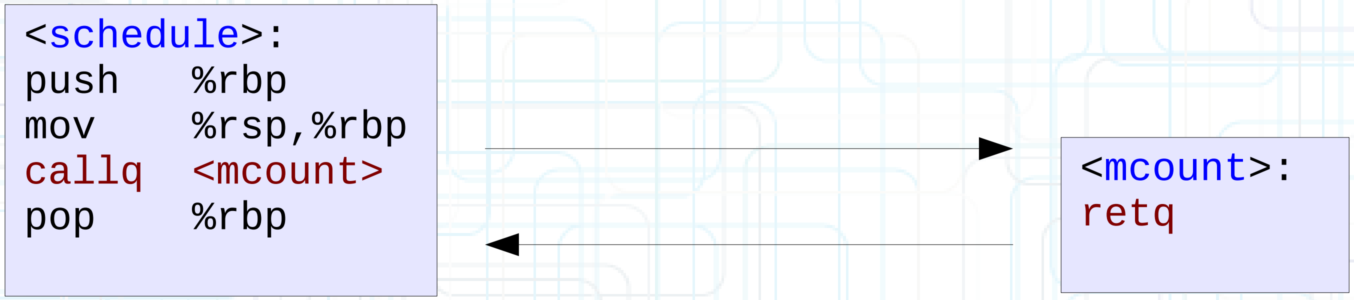
```
<schedule>:
       55                      push   %rbp
       48 8b 04 25 80 c0 0e    mov    0xffffffff810ec080,%rax
       81
       48 89 e5                mov    %rsp,%rbp
       48 8b 00                mov    (%rax),%rax
       5d                      pop    %rbp
       e9 db fa ff ff          jmpq   ffffffff810bb100 <__schedule>
       66 66 2e 0f 1f 84 00    data16 nopw %cs:0x0(%rax,%rax,1)
       00 00 00 00
```

A function call
With -pg option

- Disassembled

```
<schedule>:
       55                       push   %rbp
       48 89 e5                 mov    %rsp,%rbp
       e8 37 2e 00 00           callq  ffffffff810f7430 <mcount>
       5d                       pop    %rbp
       48 8b 04 25 80 d0 15     mov    0xffffffff8115d080,%rax
       81
       48 8b 00                 mov    (%rax),%rax
       e9 96 fa ff ff           jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00        nopw   0x0(%rax,%rax,1)
```

The kernel at boot up



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
callq  <mcount>
pop    %rbp
```

```
<mcount>:
retq
```

Where’s the mcounts?

- Can’t just call the mcounts
    - too much overhead
    - retq only is 13% added overhead!
- Need to convert them to nops at boot up

  启动过程均为nop指令
- Need to know where they are

  他们在哪
- Best to find them at compile time

  最好是在编译过程就发现他们

recordmcount

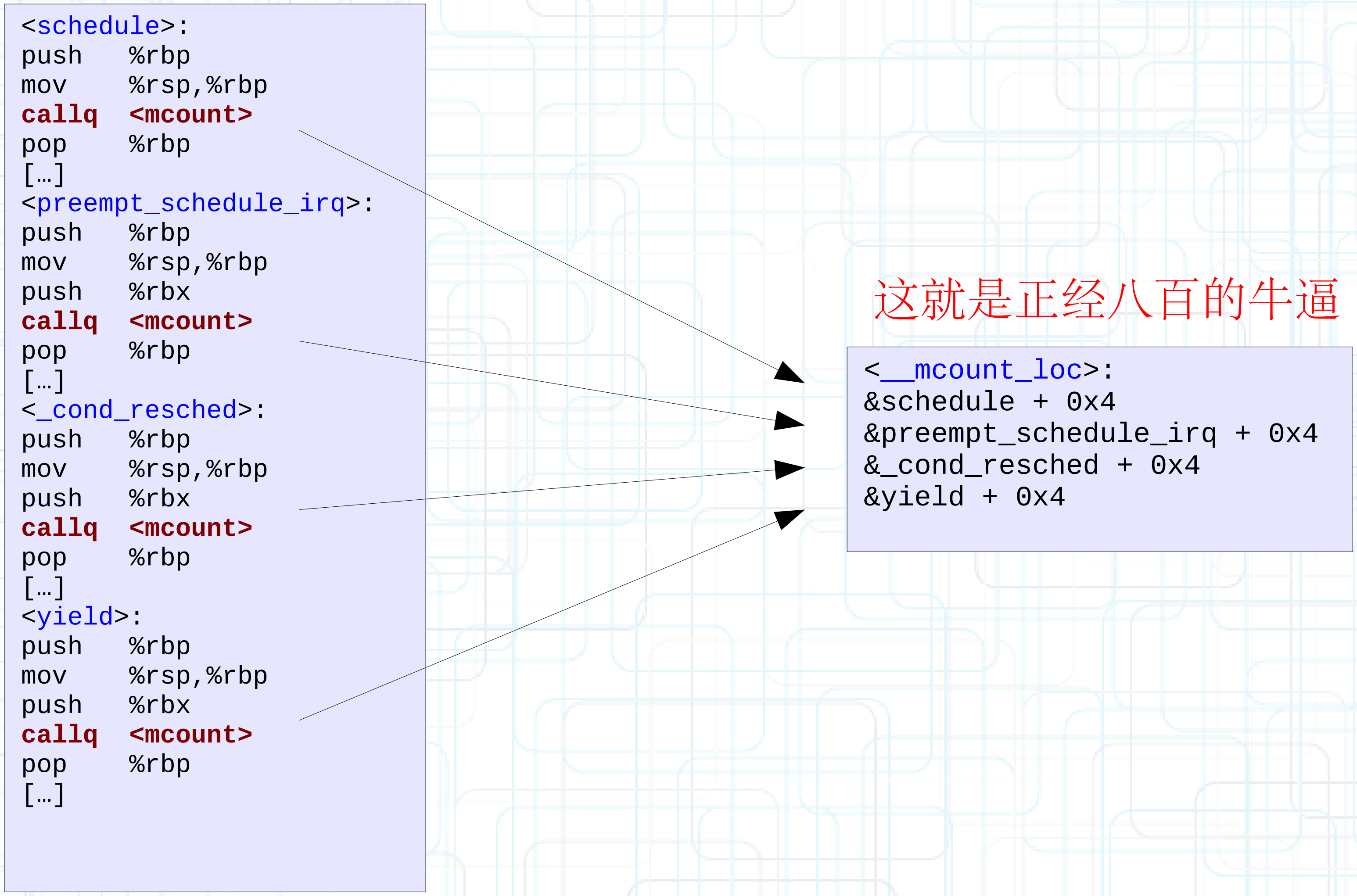
- scripts/recordmcount.c (and a perl version)
- reads the object files one at a time
- reads the relocation tables
    - finds all the calls to mcount
    - creates a table
    - links the table back into the object file
    - New section called __mcount_loc

参见内核中的section

```
__start_mcount_loc = .;        /* ffffffff83dfa990 */ \
KEEP(*(__mcount_loc))          \
KEEP(*(__patchable_function_entries))  \
__stop_mcount_loc = .;    /* ffffffff83e4a800 */ \
```

recordmcount

kernel/sched/core.o:



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
callq  <mcount>
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
```

这就是正经八百的牛逼

```
<__mcount_loc>:
&schedule + 0x4
&preempt_schedule_irq + 0x4
&_cond_resched + 0x4
&yield + 0x4
```

recordmcount

kernel/sched/core.o:

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
callq  <mcount>
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<__mcount_loc>:
&schedule + 0x4
&preempt_schedule_irq + 0x4
&_cond_resched + 0x4
&yield + 0x4
```

将其添加到.o文件中

Linker Magic

- vmlinux.lds
  - include/linux/vmlinux.lds.h
  - arch/x86/kernel/vmlinux.lds.S
- Magic variables
  - __start_mcount_loc
  - __stop_mcount_loc

```
#ifdef CONFIG_FTRACE_MCOUNT_RECORD
#define MCOUNT_REC()    . = ALIGN(8);                          \
                        VMLINUX_SYMBOL(__start_mcount_loc) = .; \
                        *(__mcount_loc)                        \
                        VMLINUX_SYMBOL(__stop_mcount_loc) = .;
#else
#define MCOUNT_REC()
#endif
```

Linker Magic

vmlinux:

kernel/sched/core.o:

```
<__mcount_loc>:
&schedule + 0x4
&preempt_schedule_irq + 0x4
&_cond_resched + 0x4
&yield + 0x4
```

mm/swap.o:

```
<__mcount_loc>:
&put_page + 0x4
&__get_page_tail + 0x4
&put_pages_list + 0x4
&get_kernel_pages + 0x4
```

fs/read_write.o:

```
<__mcount_loc>:
&new_sync_read + 0x4
&vfs_setpos + 0x4
&fixed_size_llseek + 0x4
&default_llseek + 0x4
```

...

Linker Magic

vmlinux:

```
<__start_mcount_loc>:
&schedule + 0x4
&preempt_schedule_irq + 0x4
&_cond_resched + 0x4
&yield + 0x4
&put_page + 0x4
&__get_page_tail + 0x4
&put_pages_list + 0x4
&get_kernel_pages + 0x4
&new_sync_read + 0x4
&vfs_setpos + 0x4
&fixed_size_llseek + 0x4
&default_llseek + 0x4
 [...]
<__end_mcount_loc>:
```

kernel/sched/core.o:

```
<__mcount_loc>:
&schedule + 0x4
&preempt_schedule_irq + 0x4
&_cond_resched + 0x4
&yield + 0x4
```

mm/swap.o:

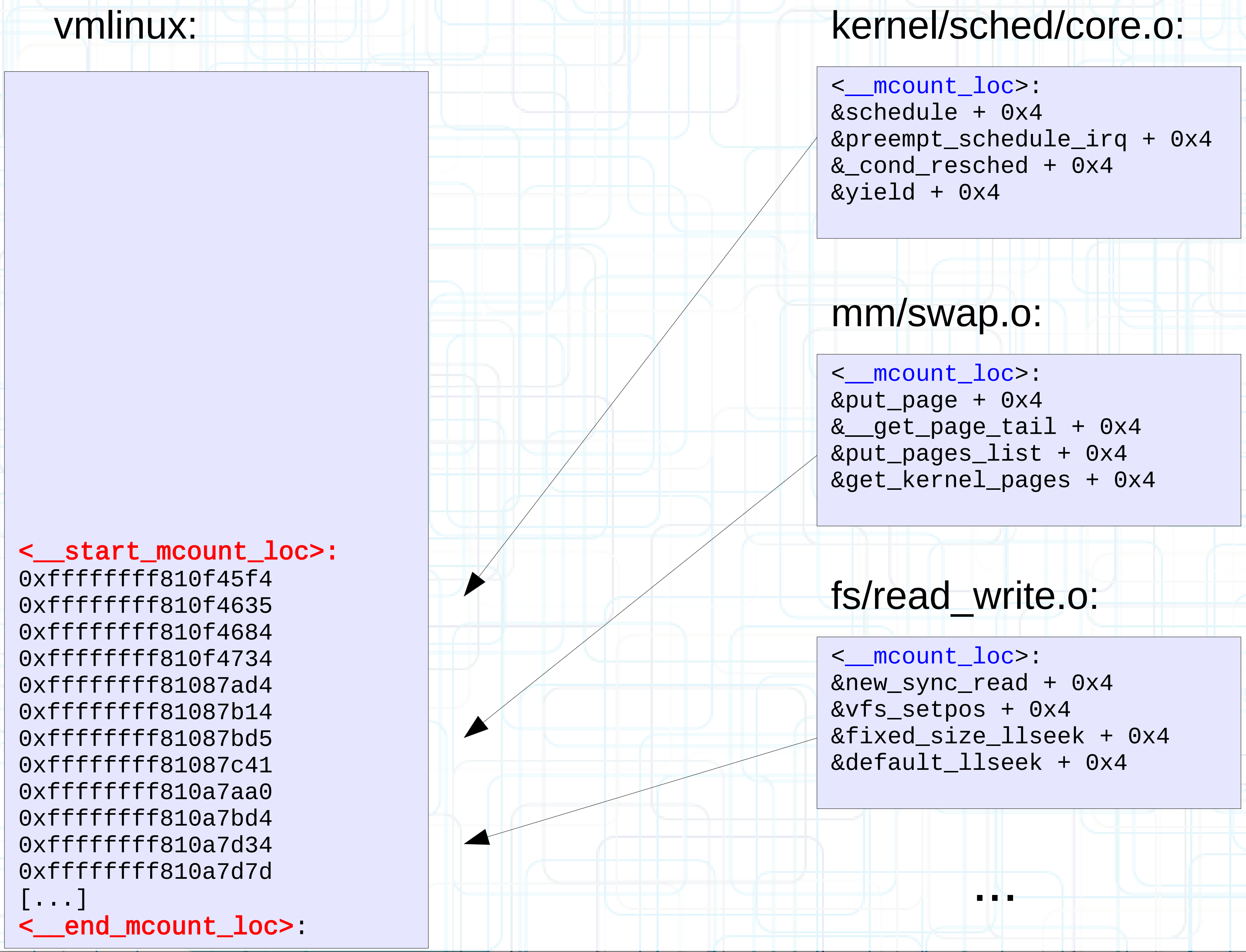
```
<__mcount_loc>:
&put_page + 0x4
&__get_page_tail + 0x4
&put_pages_list + 0x4
&get_kernel_pages + 0x4
```

fs/read_write.o:

```
<__mcount_loc>:
&new_sync_read + 0x4
&vfs_setpos + 0x4
&fixed_size_llseek + 0x4
&default_llseek + 0x4
```

...

Linker Magic



vmlinux:

```
<__start_mcount_loc>:
0xffffffff810f45f4
0xffffffff810f4635
0xffffffff810f4684
0xffffffff810f4734
0xffffffff81087ad4
0xffffffff81087b14
0xffffffff81087bd5
0xffffffff81087c41
0xffffffff810a7aa0
0xffffffff810a7bd4
0xffffffff810a7d34
0xffffffff810a7d7d
[...]
<__end_mcount_loc>:
```

kernel/sched/core.o:

```
<__mcount_loc>:
&schedule + 0x4
&preempt_schedule_irq + 0x4
&_cond_resched + 0x4
&yield + 0x4
```

mm/swap.o:

```
<__mcount_loc>:
&put_page + 0x4
&__get_page_tail + 0x4
&put_pages_list + 0x4
&get_kernel_pages + 0x4
```

fs/read_write.o:

```
<__mcount_loc>:
&new_sync_read + 0x4
&vfs_setpos + 0x4
&fixed_size_llseek + 0x4
&default_llseek + 0x4
```

...

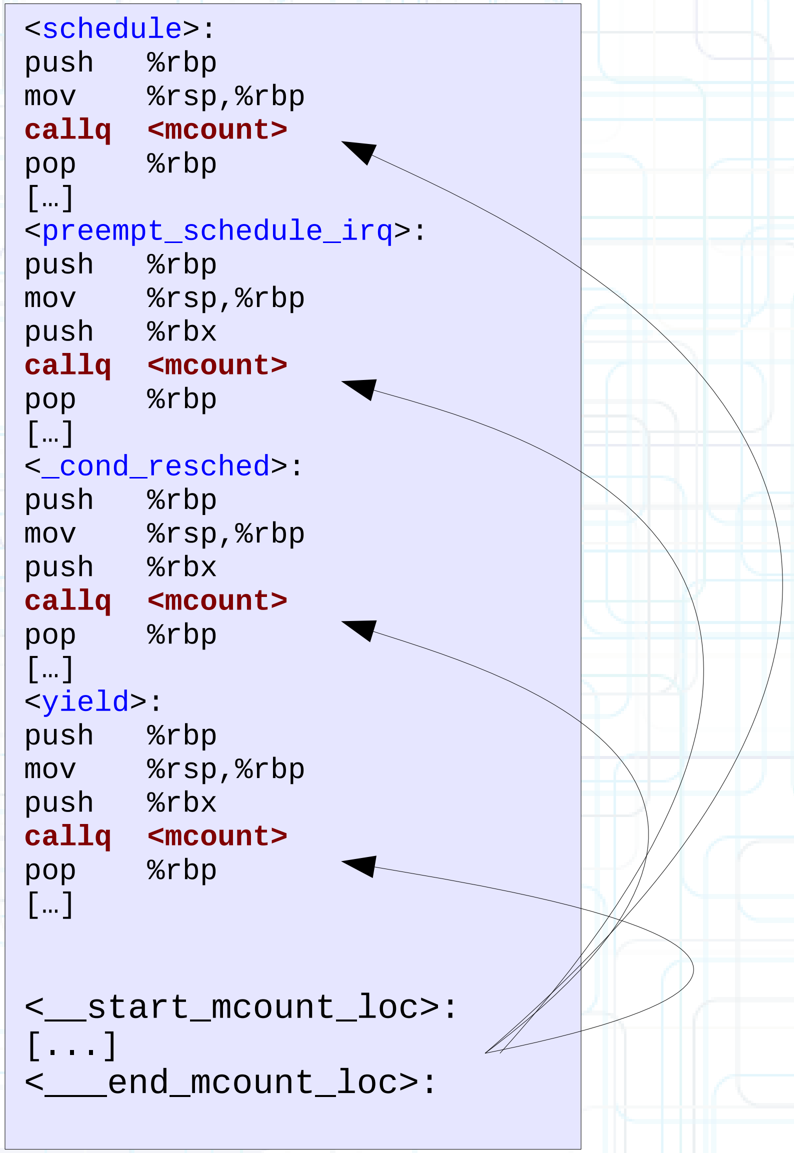
Finding Mcount

vmlinux:

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
callq  <mcount>
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]


<__start_mcount_loc>:
[...]
<___end_mcount_loc>:
```

Finding Mcount

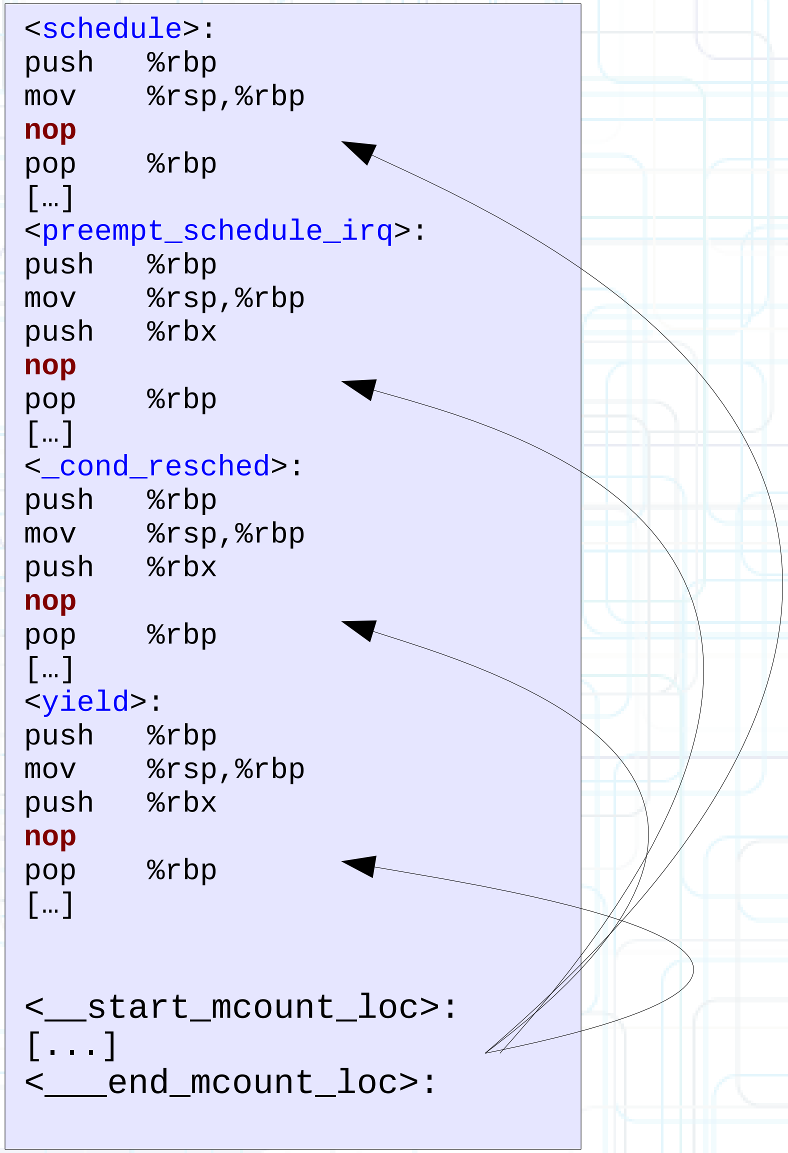
vmlinux: 当开启mcount/ftrace功能



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
callq  <mcount>
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
callq  <mcount>
pop    %rbp
[…]


<__start_mcount_loc>:
[...]
<___end_mcount_loc>:
```

Finding Mcount

vmlinux: 当关闭ftrace功能



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
nop
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]


<__start_mcount_loc>:
[...]
<___end_mcount_loc>:
```

Finding Mcount

vmlinux:

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
nop
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]


<__start_mcount_loc>:
[...]
<___end_mcount_loc>:
```

Finding Mcount

vmlinux:

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
nop
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
```

What about tracing?

使能方法

- Need a way to enable tracing

扔掉mcount section

- We threw away the mcount section
- The mcount section wasn’t enough for us
- Tracing also requires saving state

struct dyn_ftrace

```
struct dyn_ftrace {
	unsigned long		ip; /* address of mcount call-site */
	unsigned long		flags;
	struct dyn_arch_ftrace	arch;
};
```

struct dyn_ftrace

```
struct dyn_ftrace {
        unsigned long           ip; /* address of mcount call-site */
        unsigned long           flags;
        struct dyn_arch_ftrace  arch;
};

arch/x86/include/asm/ftrace.h:
struct dyn_arch_ftrace {
        /* No extra data needed for x86 */
};
```

struct dyn_ftrace

```
struct dyn_ftrace {
	unsigned long		ip; /* address of mcount call-site */
	unsigned long		flags;
	struct dyn_arch_ftrace	arch;
};

arch/powerpc/include/asm/ftrace.h:
struct dyn_arch_ftrace {
	struct module *mod;
};
```

Tracing data

- Copy from mcount_loc before deleting
- Sorted for quick lookup 排序- 为了查找
- Allocated in groups of pages
    - details out of scope for this talk
- Data reported at boot up

```
# dmesg |grep ftrace
[    0.139656] ftrace: allocating 24683 entries in 97 pages
```

    - Allocated 24,683 dyn_ftrace structures
    - Used up 97 (4K) pages to do so
    - Total of 397,312 bytes

Tracing data

vmlinux:

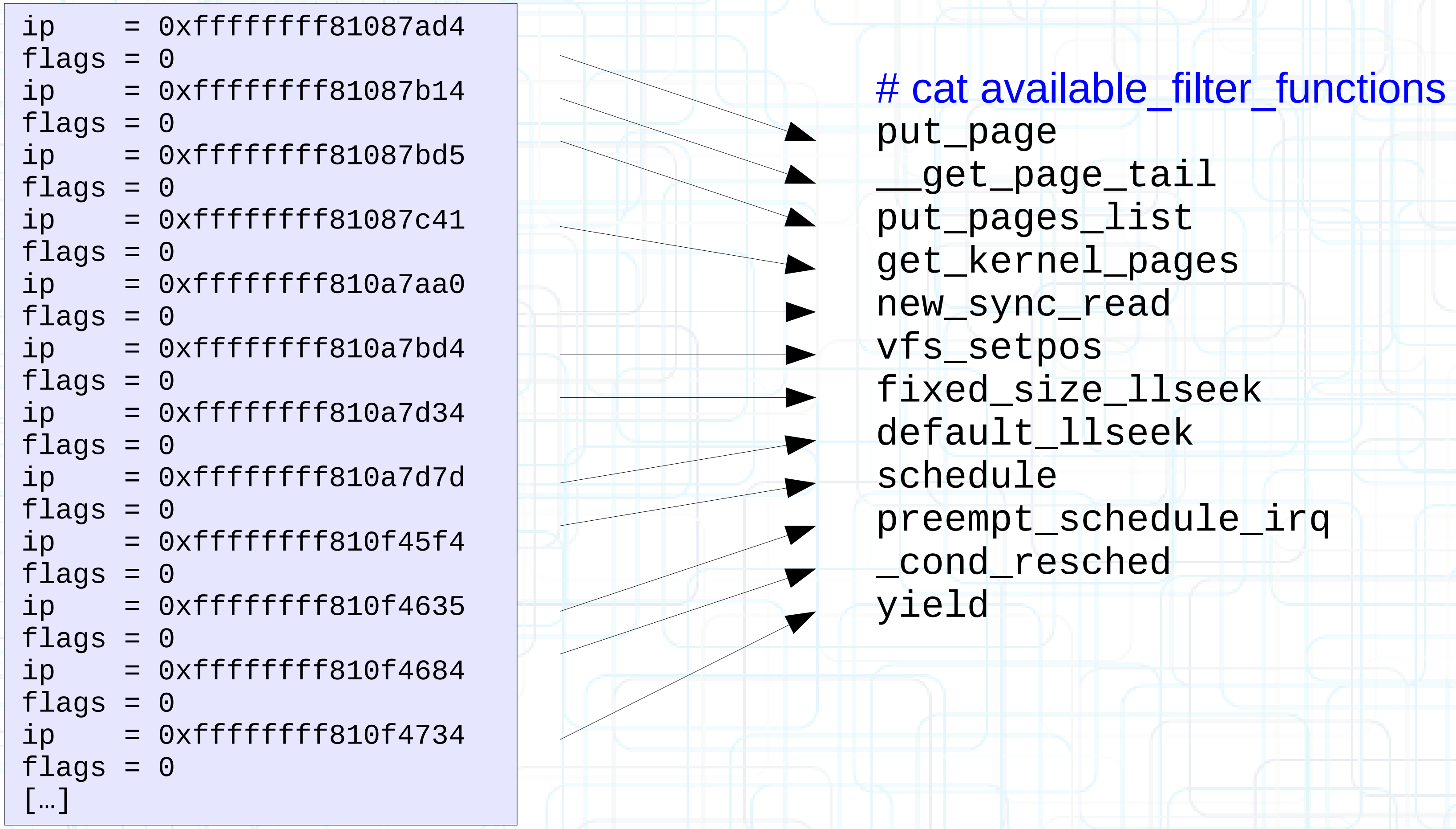
```
<schedule>:
push   %rbp
mov    %rsp,%rbp
nop
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]


<__start_mcount_loc>:
[...]
<___end_mcount_loc>:
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

Tracing data

<ftrace_pages> 可用的



```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

```
# cat available_filter_functions
put_page
__get_page_tail
put_pages_list
get_kernel_pages
new_sync_read
vfs_setpos
fixed_size_llseek
default_llseek
schedule
preempt_schedule_irq
_cond_resched
yield
```

Tracing data

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

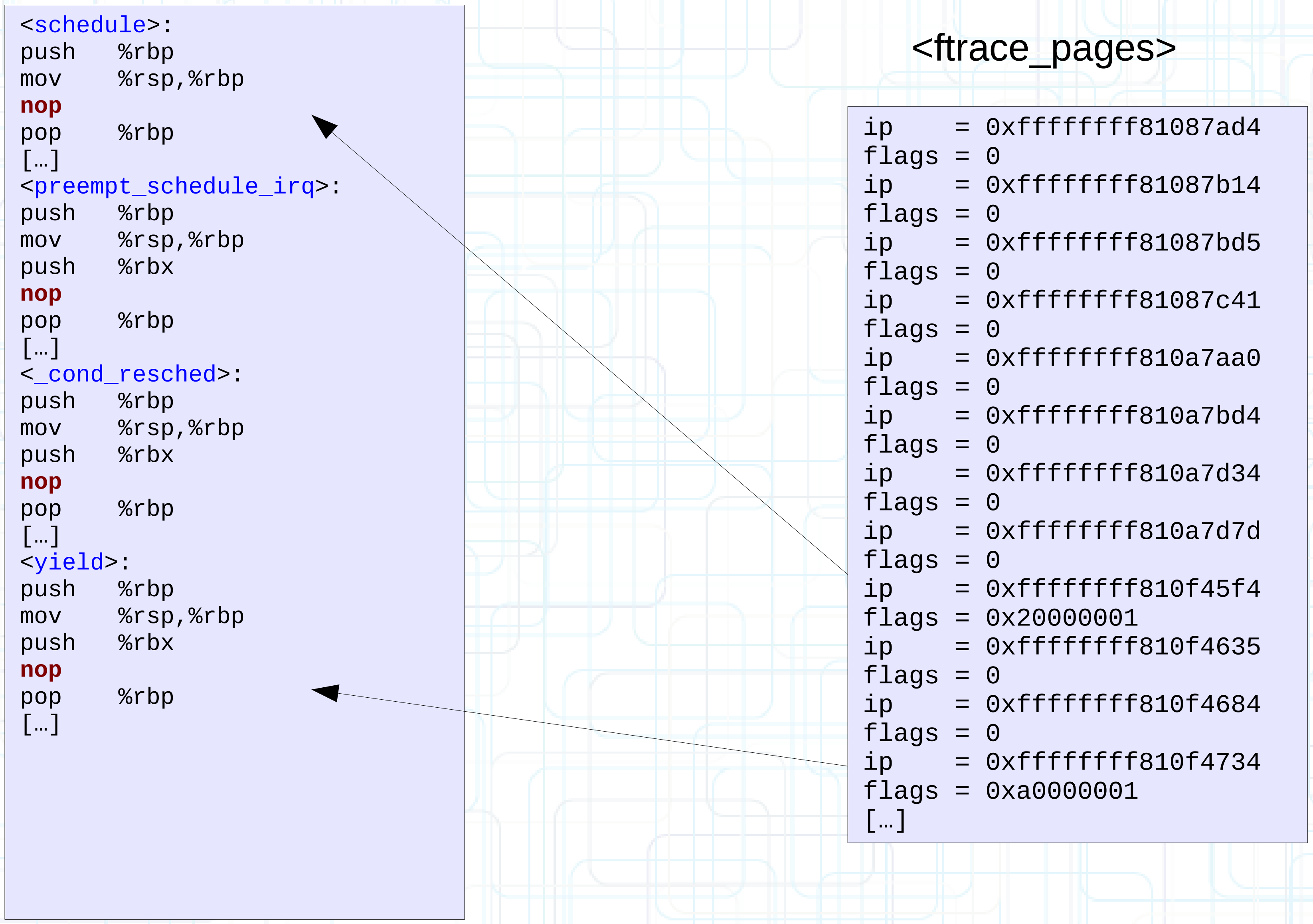
```
# echo yield > set_ftrace_filter
# echo schedule >> set_ftrace_filter
# cat set_ftrace_filter
schedule
yield
```

dyn_ftrace.flags

- First 29 bits are for counter
    - Every registered callback increments +1
- bit 29 (starts from zero) – ENABLED
- bit 30 – REGS
- bit 31 – REGS_EN

使能tracing

Enabling tracing



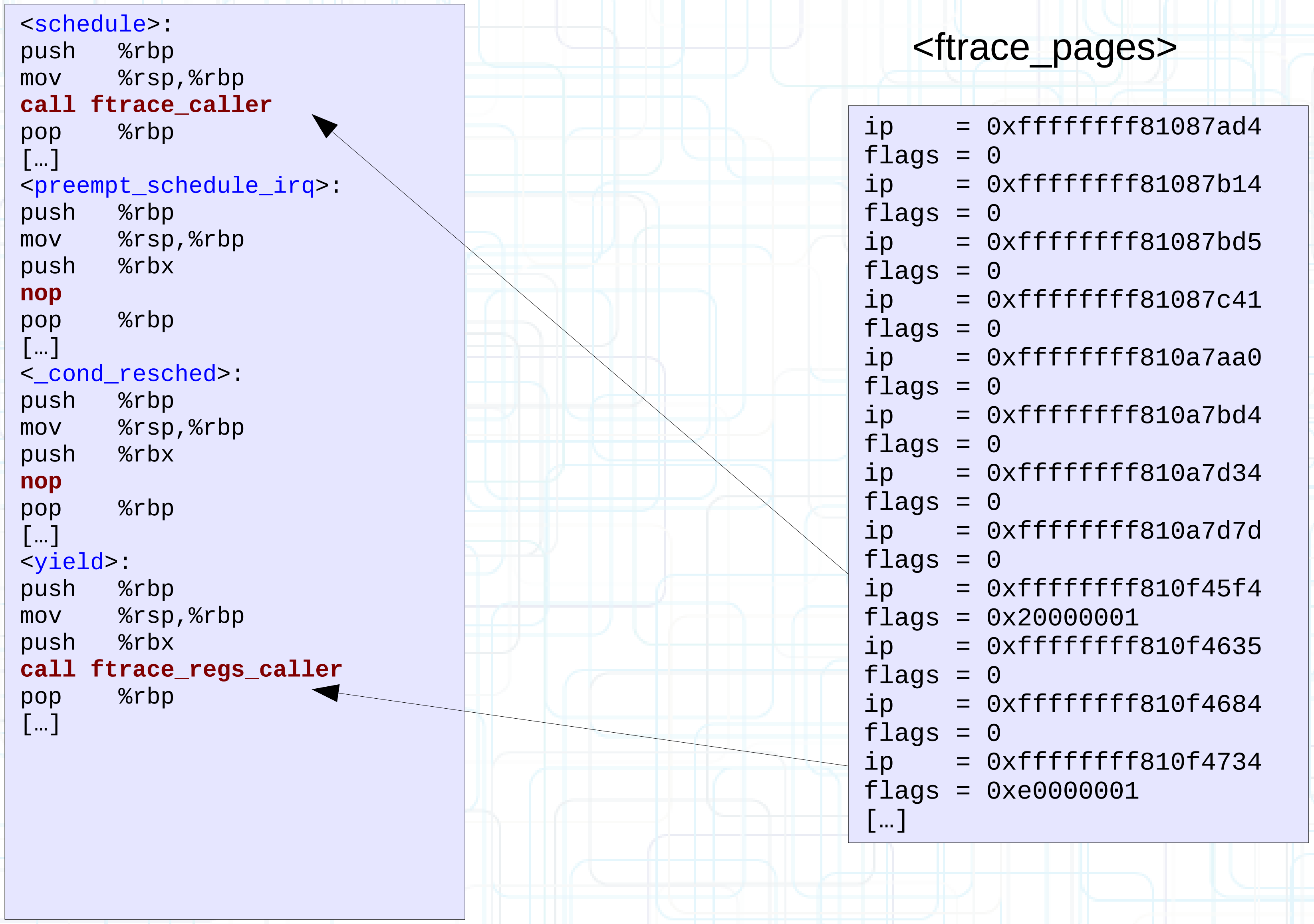
vmlinux:

```
<schedule>:
push    %rbp
mov     %rsp,%rbp
nop
pop     %rbp
[…]
<preempt_schedule_irq>:
push    %rbp
mov     %rsp,%rbp
push    %rbx
nop
pop     %rbp
[…]
<_cond_resched>:
push    %rbp
mov     %rsp,%rbp
push    %rbx
nop
pop     %rbp
[…]
<yield>:
push    %rbp
mov     %rsp,%rbp
push    %rbx
nop
pop     %rbp
[…]
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0x20000001
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0xa0000001
[…]
```

Enabling tracing



vmlinux:

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_regs_caller
pop    %rbp
[…]
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0x20000001
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0xe0000001
[…]
```

Modifying code at runtime

- Not the same as at boot up
- SMP boxes must be careful 多处理器需要注意

  其他的CPU可能执行这个代码
- Other CPUs may be executing that code

  x86具有非统一的机器指令
- x86 has non uniform machine instructions

  指令可能会跨越缓存边界
- Instructions may cross cache boundaries

Modifying code at runtime

```
<schedule>:
       55                     push   %rbp
       48 89 e5               mov    %rsp,%rbp
       0f 1f 44 00 00         nop
       5d                     pop    %rbp
       48 8b 04 25 80 d0 15   mov    0xffffffff8115d080,%rax
       81
       48 8b 00               mov    (%rax),%rax
       e9 96 fa ff ff         jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00      nopw   0x0(%rax,%rax,1)
```

Modifying code at runtime

```
<schedule>:
       55                         push   %rbp
       48 89 e5                   mov    %rsp,%rbp
       e8 37 2e 00 00             callq  ffffffff810f7430 <ftrace_caller>
       5d                         pop    %rbp
       48 8b 04 25 80 d0 15       mov    0xffffffff8115d080,%rax
       81
       48 8b 00                   mov    (%rax),%rax
       e9 96 fa ff ff             jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00          nopw   0x0(%rax,%rax,1)
```

Modifying code at runtime

CPU 0

```
<schedule>:
        55
        48 89 e5
        0f 1f 44 00 00
        5d
        48 8b 04 25 80 d0 15
        81
        48 8b 00
        e9 96 fa ff ff
        66 0f 1f 44 00 00
```

CPU 1

```
<schedule>:
  ───►  55
        48 89 e5
        0f 1f 44 00 00
        5d
        48 8b 04 25 80 d0 15
        81
        48 8b 00
        e9 96 fa ff ff
        66 0f 1f 44 00 00
```

```
0f 1f 44 00 00 nop
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Modifying code at runtime

CPU 0

```
<schedule>:
        55
        48 89 e5
        e8 37 2e 00 00
        5d
        48 8b 04 25 80 d0 15
        81
        48 8b 00
        e9 96 fa ff ff
        66 0f 1f 44 00 00
```

CPU 1

```
<schedule>:
        55
    →   48 89 e5
        0f 1f 44 00 00
        5d
        48 8b 04 25 80 d0 15
        81
        48 8b 00
        e9 96 fa ff ff
        66 0f 1f 44 00 00
```

```
0f 1f 44 00 00 nop
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Modifying code at runtime

CPU 0

```
<schedule>:
        55
        48 89 e5
        e8 37 2e 00 00
        5d
        48 8b 04 25 80 d0 15
        81
        48 8b 00
        e9 96 fa ff ff
        66 0f 1f 44 00 00
```

CPU 1

```
<schedule>:
        55
        48 89 e5
   ──►  0f 1f 2e 00 00  错误指令
        5d
        48 8b 04 25 80 d0 15
        81
        48 8b 00
        e9 96 fa ff ff
        66 0f 1f 44 00 00
```

```
0f 1f 44 00 00 nop
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

0f 1f 2e 00???

```
0f 1f 2e 00 00
```

```
0f 1f 44 00 00 nop
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

0f 1f 2e 00???

- BOOM!
- CRASH!
- GENERAL PROTECTION FAULT!
- REBOOT! 重启

How to go from this

```
<schedule>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       0f 1f 44 00 00          nop
       5d                      pop    %rbp
       48 8b 04 25 80 d0 15    mov    0xffffffff8115d080,%rax
       81
       48 8b 00                mov    (%rax),%rax
       e9 96 fa ff ff          jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
0f 1f 44 00 00 nop
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

to this?

```
<schedule>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       e8 37 2e 00 00          callq  ffffffff810f7430 <ftrace_caller>
       5d                      pop    %rbp
       48 8b 04 25 80 d0 15    mov    0xffffffff8115d080,%rax
       81
       48 8b 00                mov    (%rax),%rax
       e9 96 fa ff ff          jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
0f 1f 44 00 00 nop
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Breakpoints

使用断点解决这个问题
2021年5月12日 rtoax

Breakpoints

```
<schedule>:
       55                     push   %rbp
       48 89 e5               mov    %rsp,%rbp
       0f 1f 44 00 00         nop
       5d                     pop    %rbp
       48 8b 04 25 80 d0 15   mov    0xffffffff8115d080,%rax
       81
       48 8b 00               mov    (%rax),%rax
       e9 96 fa ff ff         jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00      nopw   0x0(%rax,%rax,1)
```

```
0f 1f 44 00 00 nop
cc 1f 44 00 00 <bp>nop
cc 37 2e 00 00 <bp>callq ffffffff810f7430 <ftrace_caller>
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Breakpoints

```
<schedule>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       cc 1f 44 00 00        <bp>nop
       5d                      pop    %rbp
       48 8b 04 25 80 d0 15    mov    0xffffffff8115d080,%rax
       81
       48 8b 00                mov    (%rax),%rax
       e9 96 fa ff ff          jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
0f 1f 44 00 00 nop
cc 1f 44 00 00 <bp>nop
cc 37 2e 00 00 <bp>callq ffffffff810f7430 <ftrace_caller>
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Breakpoints

```
<schedule>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       cc 37 2e 00 00        <bp>callq  ffffffff810f7430 <ftrace_caller>
       5d                      pop    %rbp
       48 8b 04 25 80 d0 15    mov    0xffffffff8115d080,%rax
       81
       48 8b 00                mov    (%rax),%rax
       e9 96 fa ff ff          jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
0f 1f 44 00 00 nop
cc 1f 44 00 00 <bp>nop
cc 37 2e 00 00 <bp>callq ffffffff810f7430 <ftrace_caller>
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Breakpoints

```
<schedule>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       e8 37 2e 00 00          callq  ffffffff810f7430 <ftrace_caller>
       5d                      pop    %rbp
       48 8b 04 25 80 d0 15    mov    0xffffffff8115d080,%rax
       81
       48 8b 00                mov    (%rax),%rax
       e9 96 fa ff ff          jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
0f 1f 44 00 00 nop
cc 1f 44 00 00 <bp>nop
cc 37 2e 00 00 <bp>callq ffffffff810f7430 <ftrace_caller>
e8 37 2e 00 00 callq ffffffff810f7430 <ftrace_caller>
```

Registering with Ftrace

- Call to `register_ftrace_function`()
- Requires an `ftrace_ops` descriptor
- Static ftrace_ops
    - function and function_graph
    - function probes (schedule:traceoff)
    - stack tracer
    - latency tracers
- Dynamic ftrace_ops
    - perf
    - kprobes

```
例:
//kernel/kprobes.c
enable_kprobe
    arm_kprobe
        arm_kprobe_ftrace
            __arm_kprobe_ftrace
                register_ftrace_function
```

ftrace_ops

```
struct ftrace_ops {
	ftrace_func_t			func;
	struct ftrace_ops		*next;
	unsigned long			flags;
	int __percpu			*disabled;
	void				*private;
#ifdef CONFIG_DYNAMIC_FTRACE
	struct ftrace_hash		*notrace_hash;
	struct ftrace_hash		*filter_hash;
	struct mutex			regex_lock;
#endif
};
```

ftrace_ops.flags

- ENABLED 使能
  - set by ftrace, when ops is recording
- DYNAMIC
  - set by ftrace when ops is dynamically allocated
- CONTROL
  - set by perf

```
enum {
	FTRACE_OPS_FL_ENABLED			= BIT(0),
	FTRACE_OPS_FL_DYNAMIC			= BIT(1),
	FTRACE_OPS_FL_SAVE_REGS			= BIT(2),
	FTRACE_OPS_FL_SAVE_REGS_IF_SUPPORTED	= BIT(3),
	FTRACE_OPS_FL_RECURSION_SAFE		= BIT(4),
	FTRACE_OPS_FL_STUB			= BIT(5),
	FTRACE_OPS_FL_INITIALIZED		= BIT(6),
	FTRACE_OPS_FL_DELETED			= BIT(7),
	FTRACE_OPS_FL_ADDING			= BIT(8),
	FTRACE_OPS_FL_REMOVING			= BIT(9),
	FTRACE_OPS_FL_MODIFYING			= BIT(10),
	FTRACE_OPS_FL_ALLOC_TRAMP		= BIT(11),
	FTRACE_OPS_FL_IPMODIFY			= BIT(12),
	FTRACE_OPS_FL_PID		= BIT(13),
	FTRACE_OPS_FL_RCU		= BIT(14),
	FTRACE_OPS_FL_TRACE_ARRAY		= BIT(15),
	FTRACE_OPS_FL_PERMANENT			= BIT(16),
	FTRACE_OPS_FL_DIRECT			= BIT(17),
};
```

ftrace_ops.flags

- SAVE_REGS
  - set by caller, to record regs
  - fails if saving regs is not supported
- SAVE_REGS_IF_SUPPORTED
  - set by caller, save regs if supported
  - doesn’t fail register if not supported
- RECURSION_SAFE 递归
  - If ftrace_ops.func handles recursion
  - Otherwise, ftrace will handle it

ftrace_ops.flags

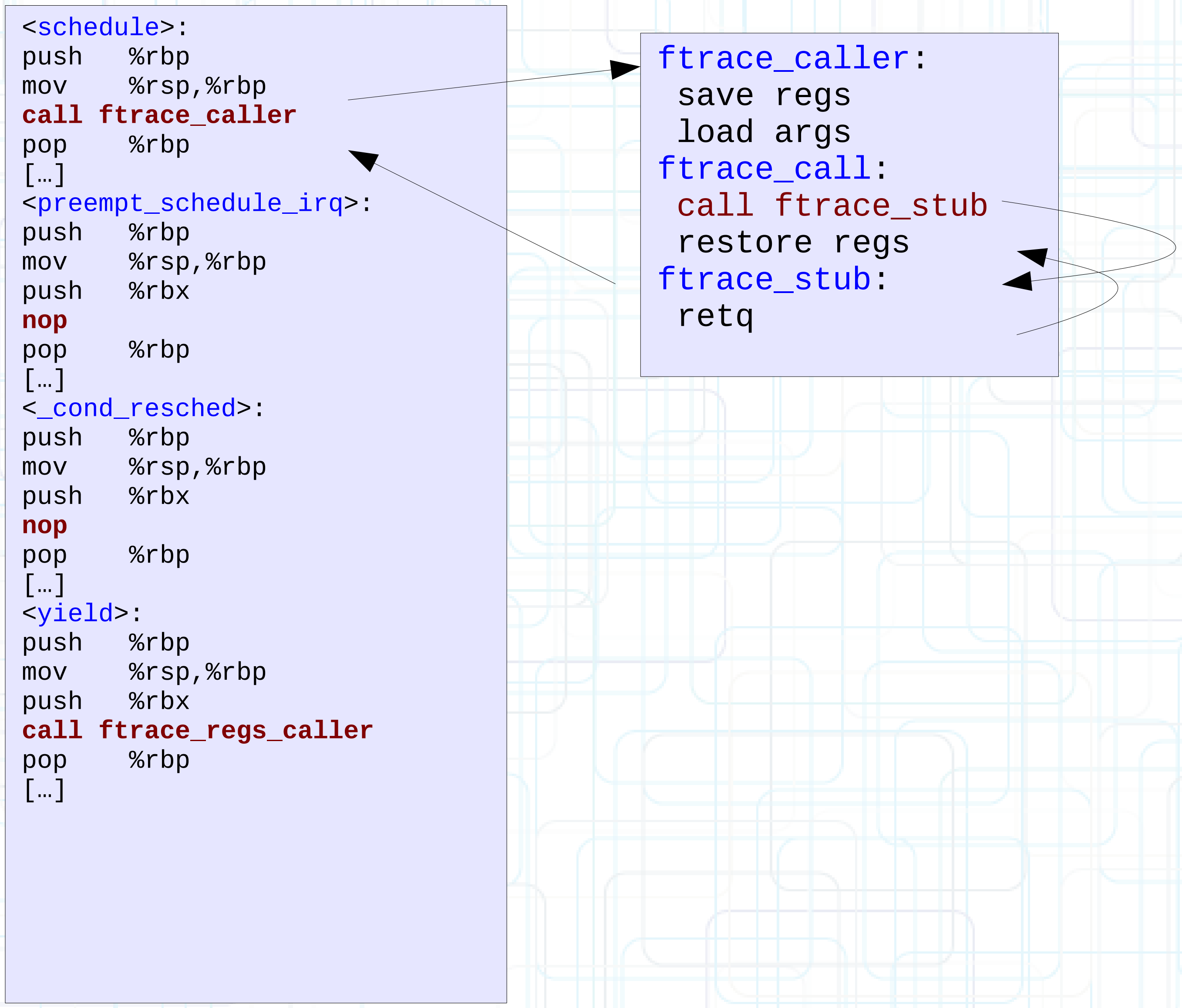
- STUB 存根
  - used by ftrace for stub functions
- INITIALIZED
  - used by ftrace when ftrace_ops is first used
- DELETED
  - ftrace_ops has been deleted
  - used by ftrace buffer instances

ftrace_ops hashes

- regex_lock
    - used to protect the hashes
- notrace_hash
    - what functions not to trace 不tracing的函数
    - empty means OK to trace all
- filter_hash
    - what functions to trace
    - empty means to trace all
- Functions in notrace_hash will not be traced even if they exist in filter_hash

蹦床

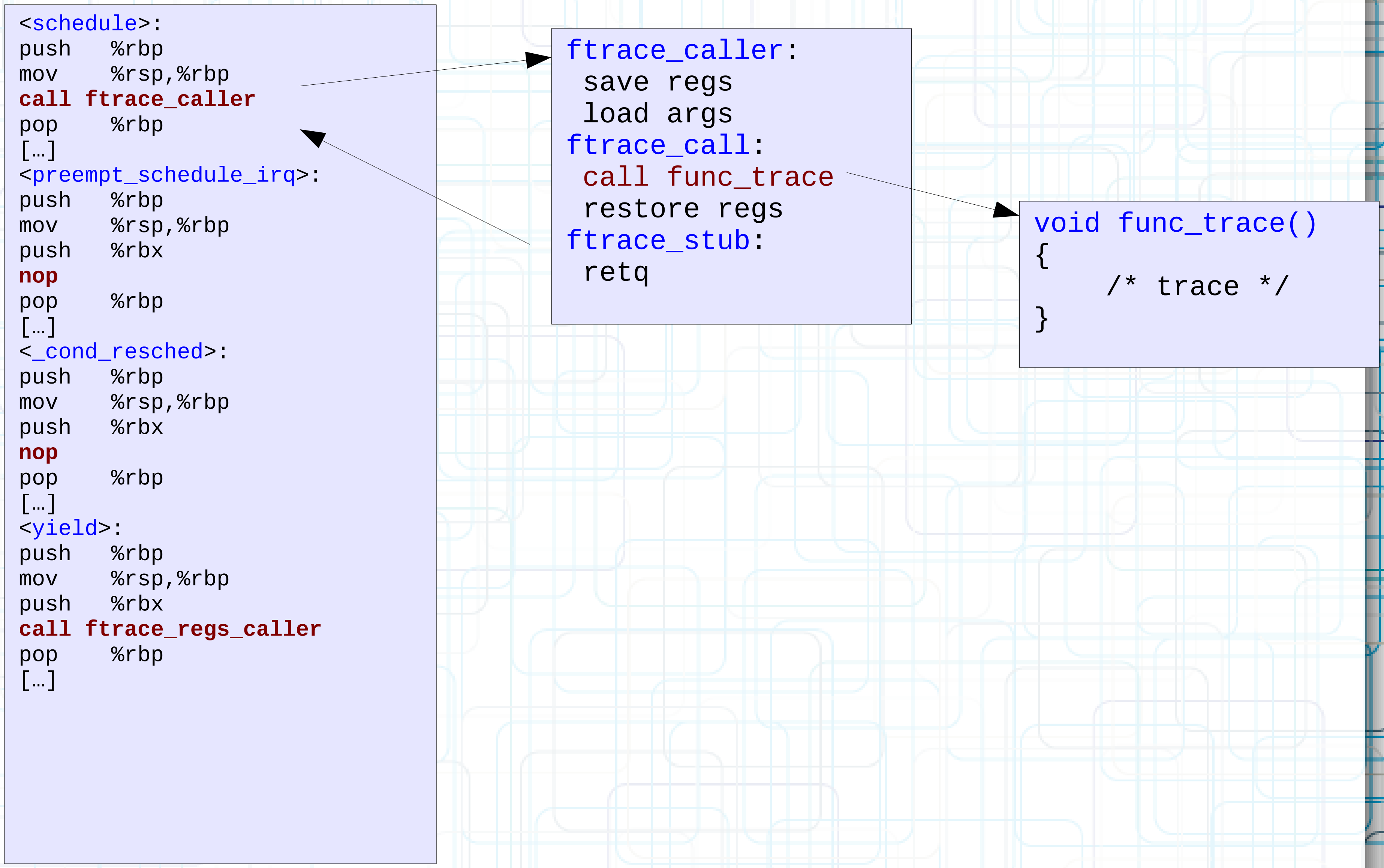
ftrace_caller trampoline



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_regs_caller
pop    %rbp
[…]
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call ftrace_stub
 restore regs
ftrace_stub:
 retq
```

ftrace_caller trampoline



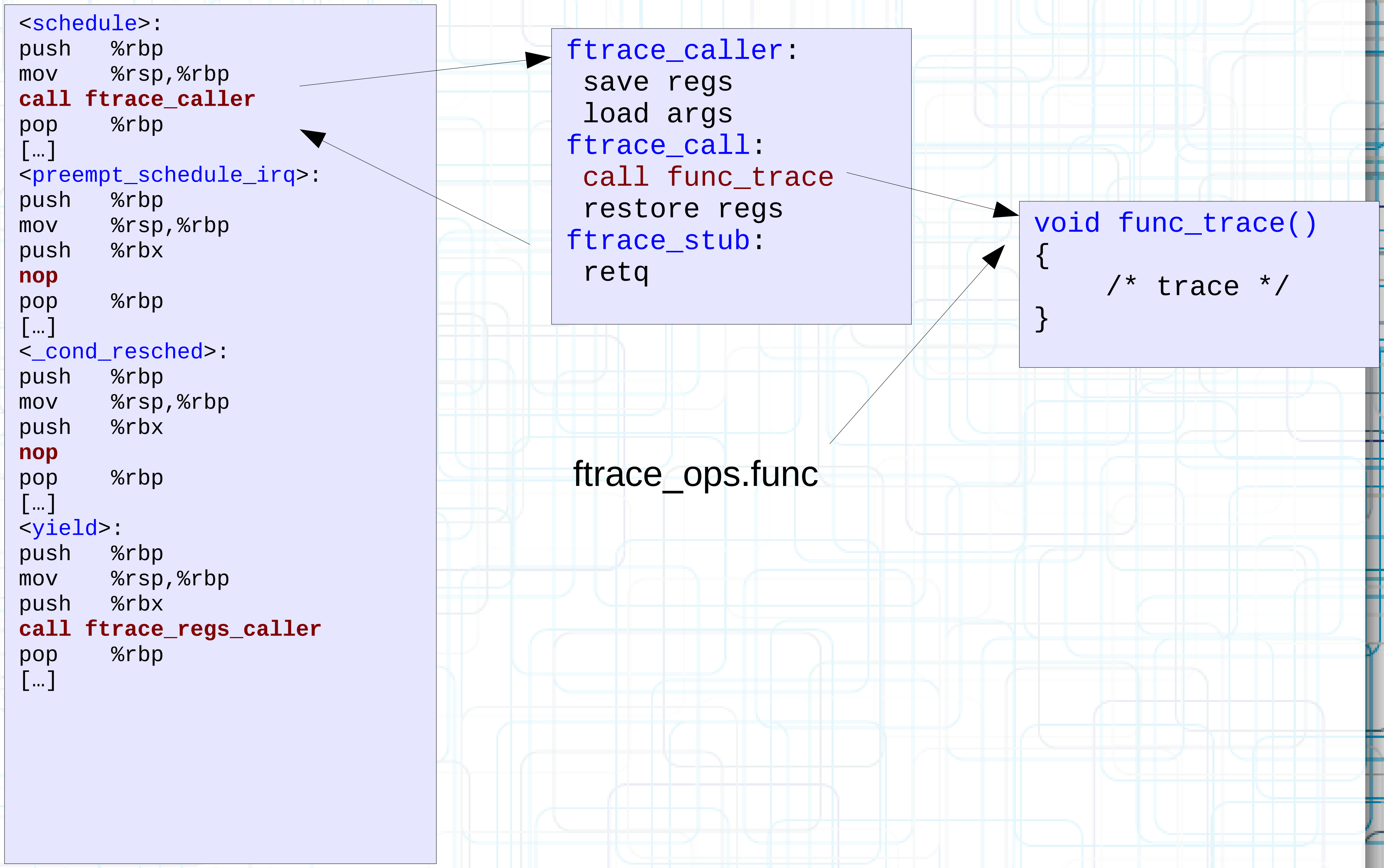
```
<schedule>:
push    %rbp
mov     %rsp,%rbp
call ftrace_caller
pop     %rbp
[…]
<preempt_schedule_irq>:
push    %rbp
mov     %rsp,%rbp
push    %rbx
nop
pop     %rbp
[…]
<_cond_resched>:
push    %rbp
mov     %rsp,%rbp
push    %rbx
nop
pop     %rbp
[…]
<yield>:
push    %rbp
mov     %rsp,%rbp
push    %rbx
call ftrace_regs_caller
pop     %rbp
[…]
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call func_trace
 restore regs
ftrace_stub:
 retq
```

```
void func_trace()
{
     /* trace */
}
```

蹦床

ftrace_caller trampoline



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_regs_caller
pop    %rbp
[…]
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call func_trace
 restore regs
ftrace_stub:
 retq
```

```
void func_trace()
{
     /* trace */
}
```

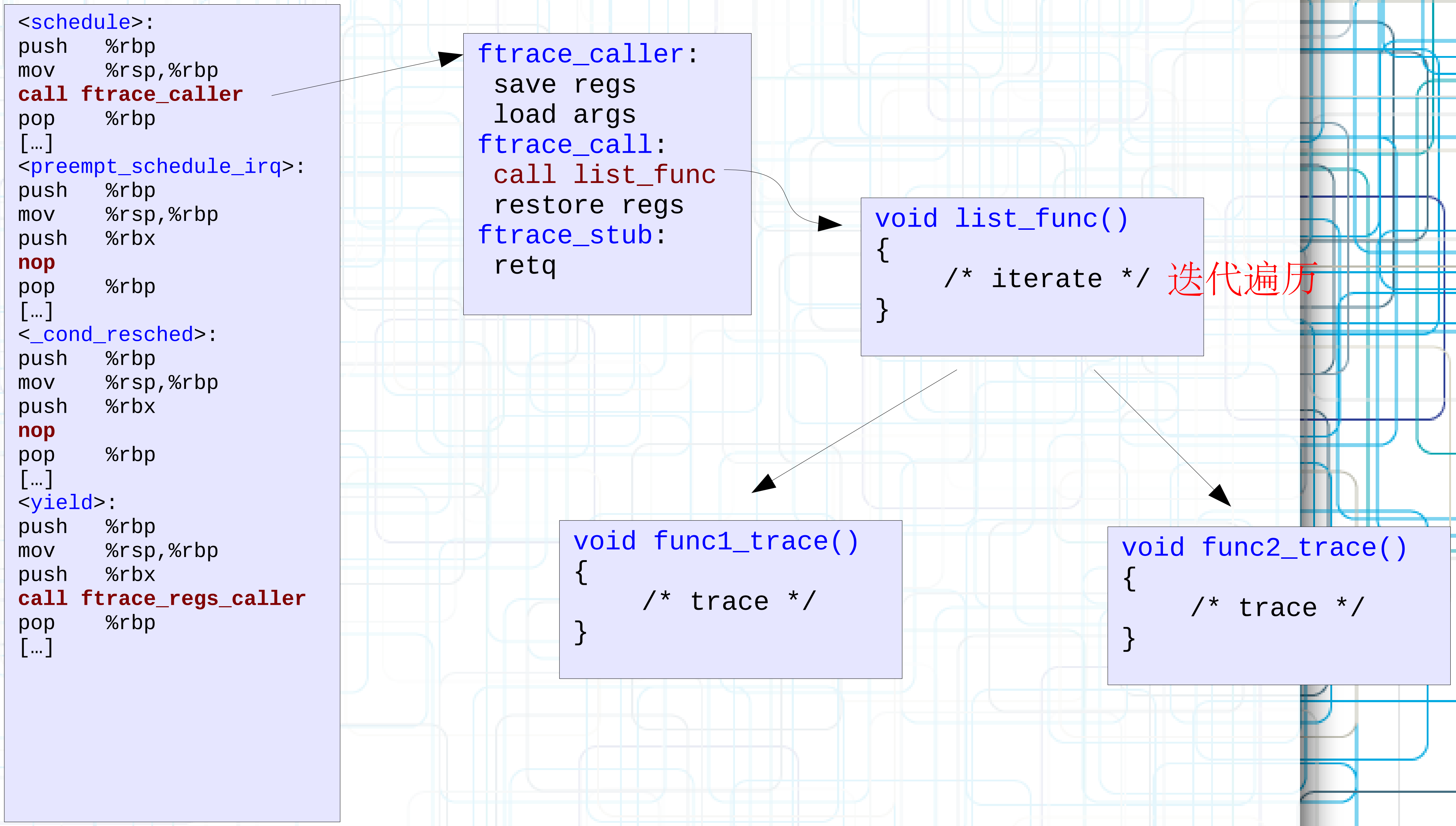
多个回调

Multiple callbacks?

- Direct call works fine
- Multiple calls requires a list operation
- All functions being traced will call the list function

多个回调函数

Multiple callbacks?



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_regs_caller
pop    %rbp
[…]
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call list_func
 restore regs
ftrace_stub:
 retq
```

```
void list_func()
{
     /* iterate */
}
```

迭代遍历

```
void func1_trace()
{
     /* trace */
}
```

```
void func2_trace()
{
     /* trace */
}
```

Example

function tracer->all functions
perf-> schedule

- Run function tracer on all functions
- Run perf on just the scheduler

function tracer->all functions
perf-> schedule

Example

Want to trace do_IRQ()?

list_func()

Nope

perf

Yep!

function_tracer

function tracer->all functions
perf-> schedule

Example

Want to trace raw_spin_lock()?

list_func()

Nope

perf

Yep!

function_tracer

function tracer->all functions
perf-> schedule

Example

Want to trace rcu_read_lock()?

list_func()

Nope

perf

Yep!

function_tracer

function tracer->all functions
perf-> schedule

Example

Want to trace rcu_read_unlock()?

list_func()

Nope

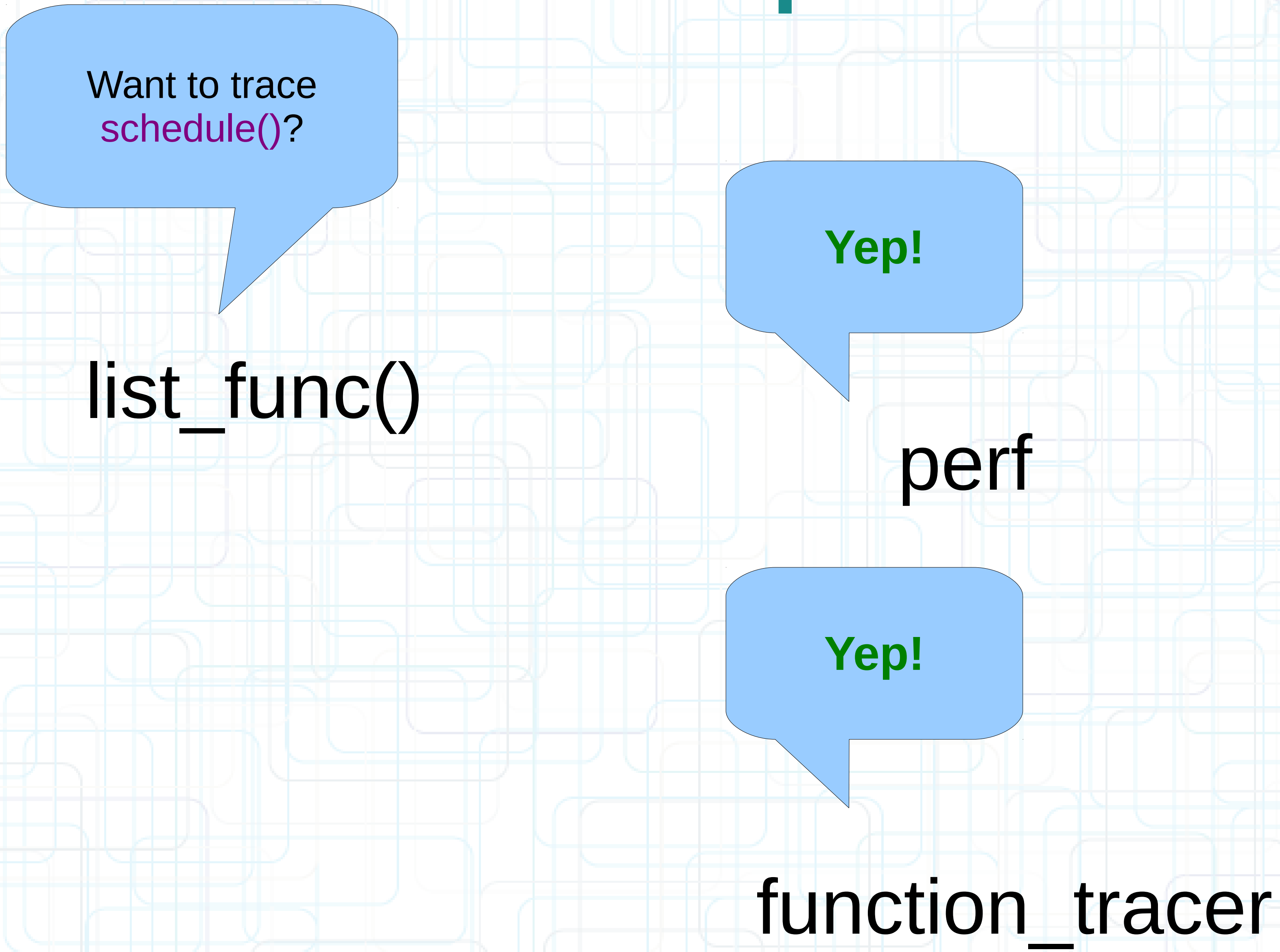
perf

Yep!

function_tracer

function tracer->all functions
perf-> schedule

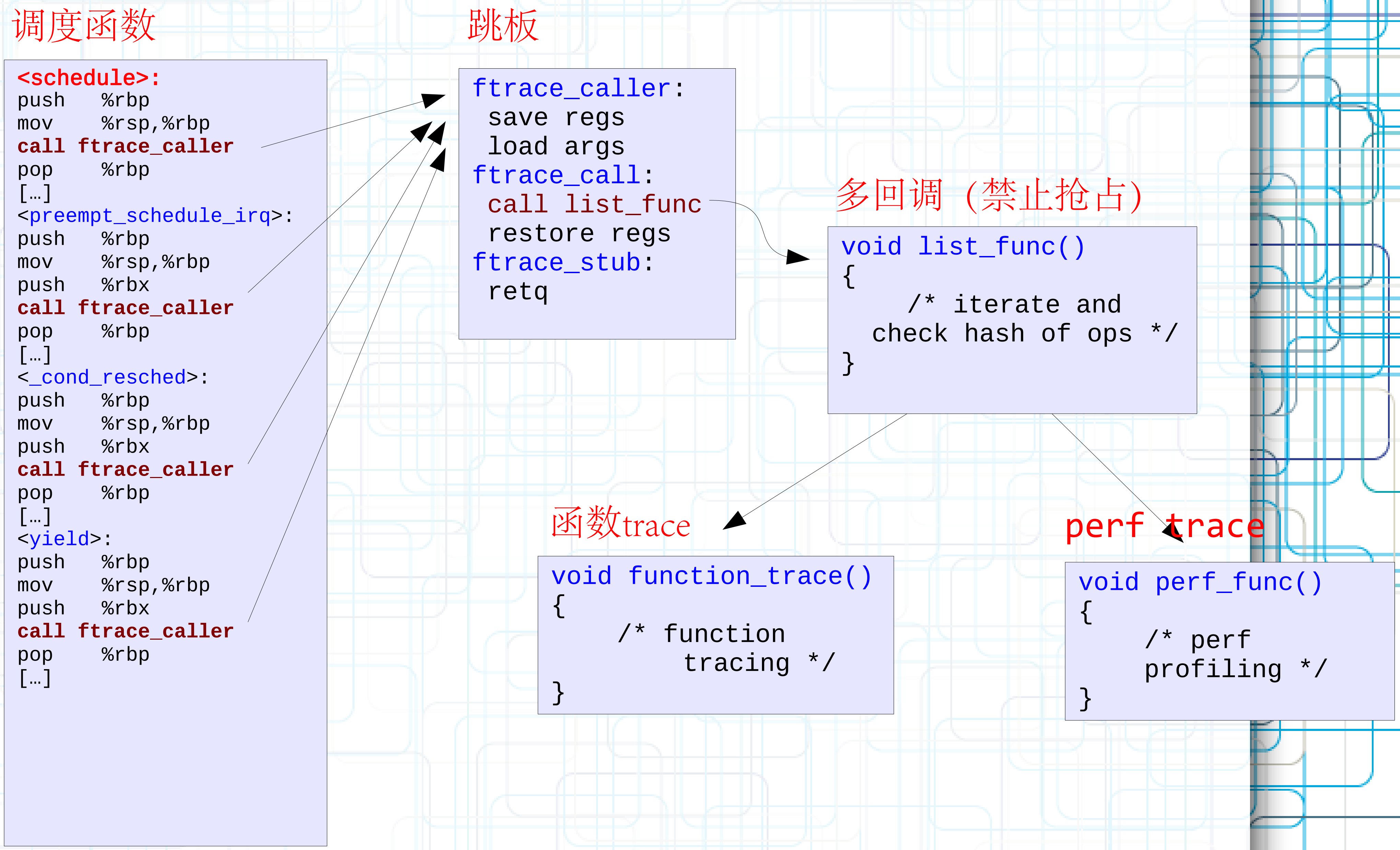
Example



function tracer->all functions

perf-> schedule

Multiple callbacks?



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_caller
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_caller
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_caller
pop    %rbp
[…]
```

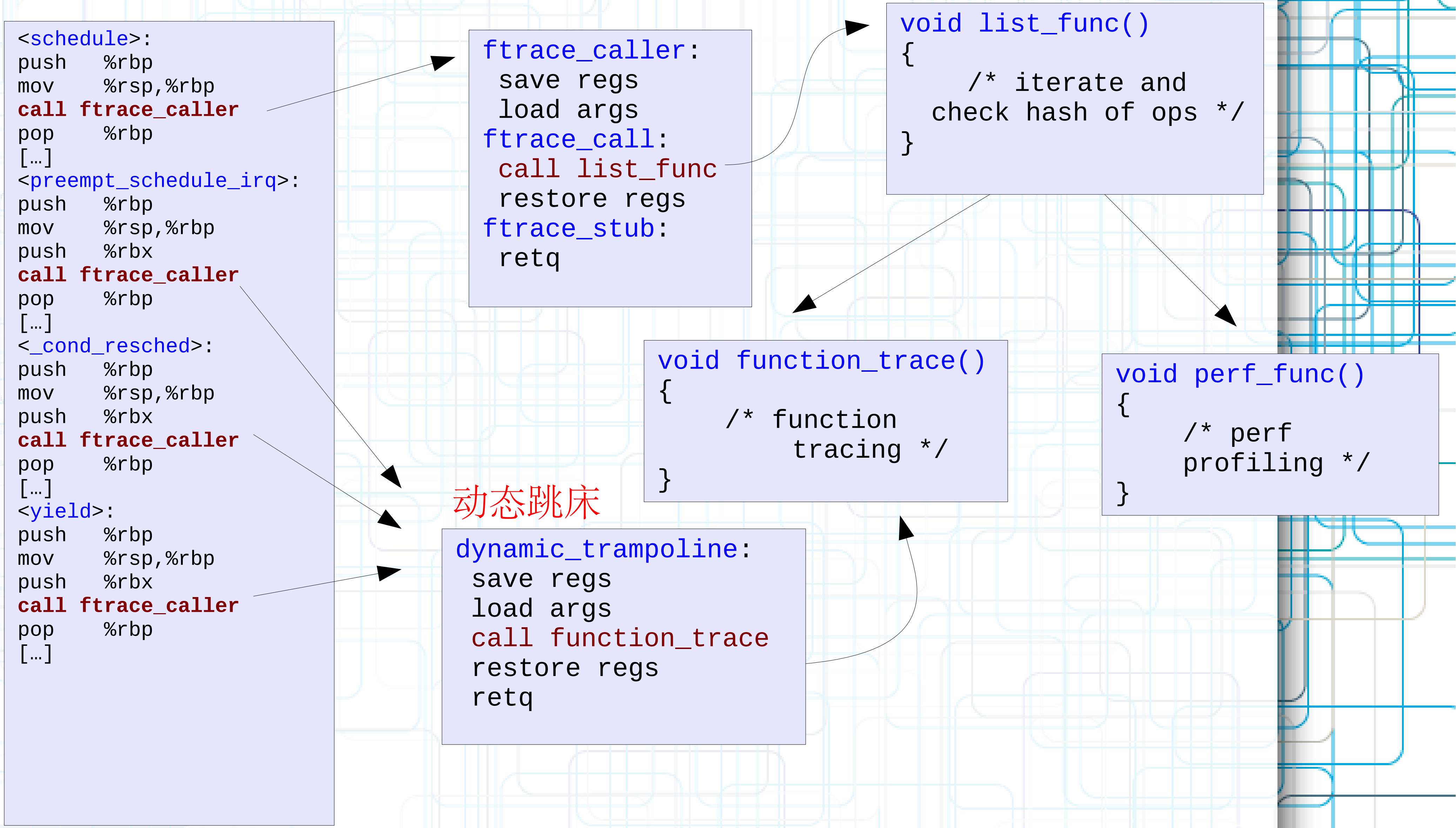
```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call list_func
 restore regs
ftrace_stub:
 retq
```

```
void list_func()
{
    /* iterate and
  check hash of ops */
}
```

```
void function_trace()
{
    /* function
          tracing */
}
```

```
void perf_func()
{
    /* perf
    profiling */
}
```

Multiple callbacks?



```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_caller
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_caller
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_caller
pop    %rbp
[…]
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call list_func
 restore regs
ftrace_stub:
 retq
```

```
void list_func()
{
    /* iterate and
  check hash of ops */
}
```

```
void function_trace()
{
    /* function
         tracing */
}
```

```
void perf_func()
{
    /* perf
    profiling */
}
```

```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```

Problems with Dynamic Trampolines

- When can you free them?
- When are they not in use?
- Would RCU help?

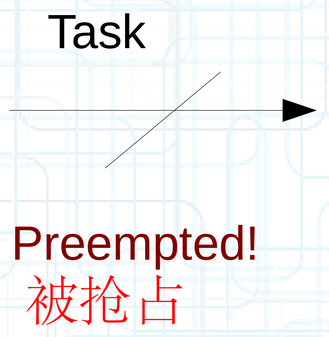
动态跳床的问题
1. 什么时候free？
2. 什么时候不被使用？
3. RCU有帮助吗？

Dynamic trampolines

Task

```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```

Dynamic trampolines



```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```

What about dynamic ftrace_ops

动态和静态 ftrace_ops 是有区别的

- Remember, there’s a difference between dynamic and static ftrace_ops

ftrace 检测 ftrace_ops 动态与否

- Ftrace detects ftrace_ops that are dymanic

总是使用 list function

- Always uses the list function
  - it disables preemption 禁止抢占
  - and is static list function 是静态的

Dynamic ftrace_ops

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_caller
pop    %rbp
[…]
<preempt_schedule_irq>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<_cond_resched>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
nop
pop    %rbp
[…]
<yield>:
push   %rbp
mov    %rsp,%rbp
push   %rbx
call ftrace_regs_caller
pop    %rbp
[…]
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call list_func
 restore regs
ftrace_stub:
 retq
```

静态的，禁止抢占的

```
void list_func()
{
 preempt_disable_notrace();
     /* iterate */
 preempt_enable_notrace();
}
```

```
void dynamic_ops_func()
{
     /* trace */
}
```

```
#define preempt_disable_notrace() \
do { \
     __preempt_count_inc(); \
     barrier(); \
} while (0)
```

Knowing when to free

- If there was a way to know no more tasks were on the trampoline or function
- There will be a way in coming 3.18

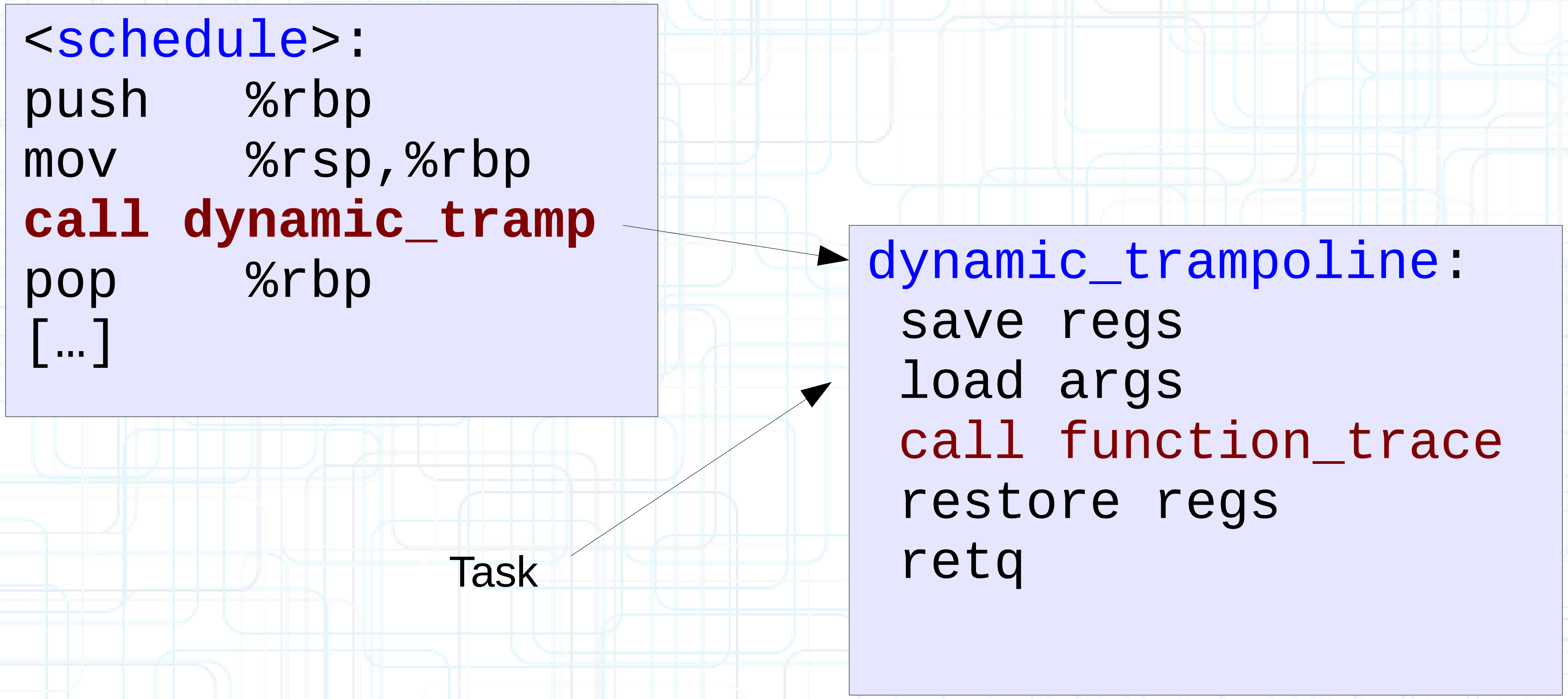
当有一种方法可以知道没有tasks在蹦床或者函数时;

call_rcu_tasks()

- Call a function after all tasks
  - have voluntarily scheduled 自愿调度
  - in userspace 用户空间
  - are idle idle任务

动态蹦床

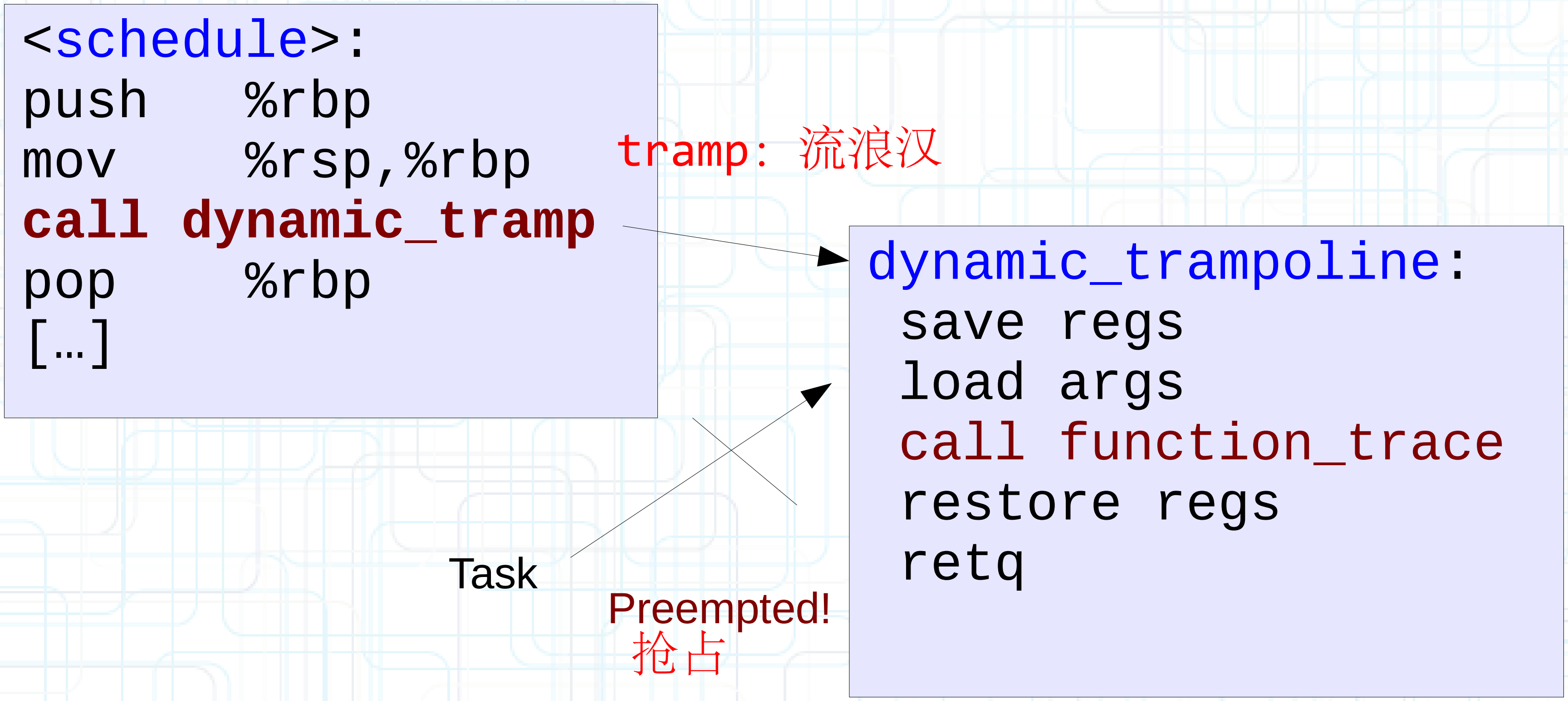
Dynamic trampolines



```
<schedule>:
push    %rbp
mov     %rsp,%rbp
call dynamic_tramp
pop     %rbp
[…]
```

```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```

Dynamic trampolines



```
<schedule>:
push    %rbp
mov     %rsp,%rbp
call dynamic_tramp
pop     %rbp
[…]
```

tramp: 流浪汉

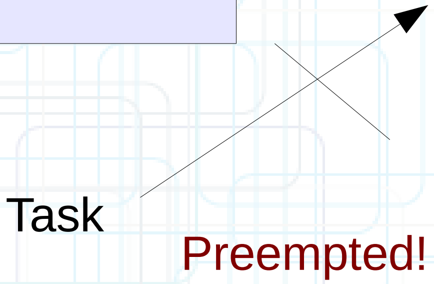
```
dynamic_trampoline:
  save regs
  load args
  call function_trace
  restore regs
  retq
```

Dynamic trampolines

```
<schedule>:
push    %rbp
mov     %rsp,%rbp
call ftrace_stub
pop     %rbp
[…]
```

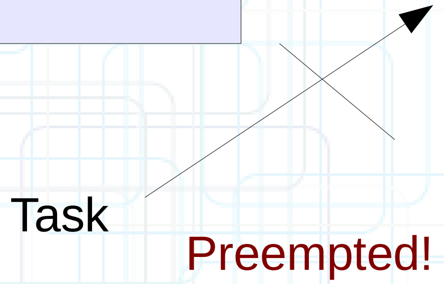
stub: 存根

```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```



Dynamic trampolines

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_stub
pop    %rbp
[…]
```



```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```

call_rcu_task()

Dynamic trampolines

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_stub
pop    %rbp
[…]
```

```
dynamic_trampoline:
 save regs
 load args
 call function_trace
 restore regs
 retq
```

Task

- Voluntary schedule
- In idle
- In userspace

call_rcu_task()

Dynamic trampolines

```
<schedule>:
push   %rbp
mov    %rsp,%rbp
call ftrace_stub
pop    %rbp
[…]
```

```
dynamic_trampoline:
  save regs
  load args
  call function_trace
  restore regs
  retq
```

Task

- Voluntary schedule
- In idle
- In userspace

fentry

mcount不能记录参数

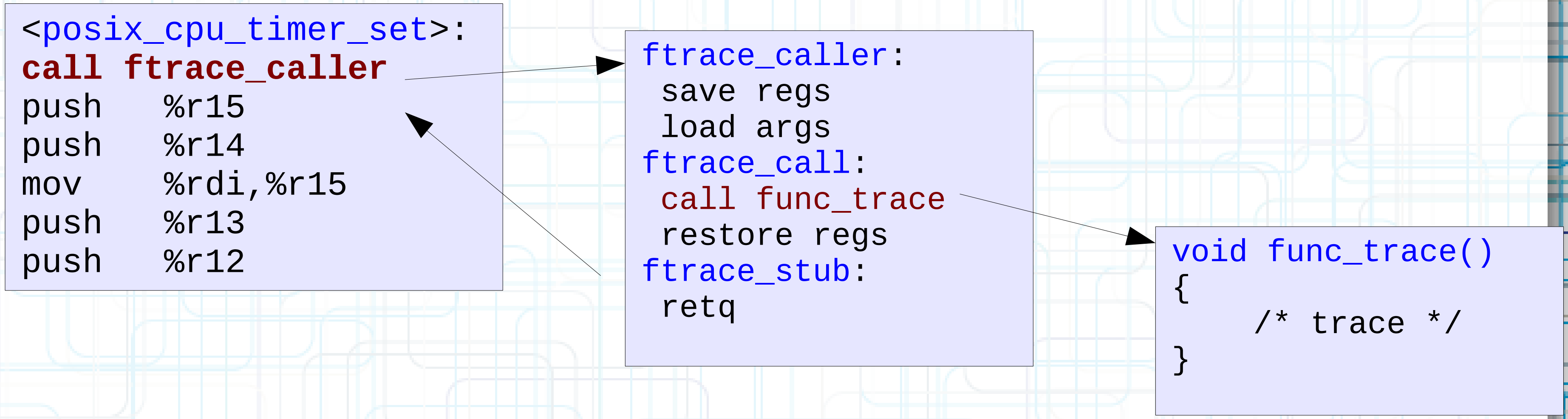
- mcount can’t record parameters
- New feature of gcc
  - starting with gcc 4.6.0
    - Added by Andi Kleen
  - for x86_64 only (for now)
- gcc -pg -mfentry   -mfentry

```
<schedule>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       e8 37 2e 00 00          callq  ffffffff810f7430 <mcount>
       5d                      pop    %rbp
       48 8b 04 25 80 d0 15    mov    0xffffffff8115d080,%rax
       81
       48 8b 00                mov    (%rax),%rax
       e9 96 fa ff ff          jmpq   ffffffff810f40a0 <__schedule>
       66 0f 1f 44 00 00       nopw   0x0(%rax,%rax,1)
```

```
<posix_cpu_timer_set>:
       55                      push   %rbp
       48 89 e5                mov    %rsp,%rbp
       41 57                   push   %r15
       41 56                   push   %r14
       41 55                   push   %r13
       41 54                   push   %r12
       53                      push   %rbx
       48 83 ec 30             sub    $0x30,%rsp
       e8 1a 81 0b 00          callq  ffffffff810f7430 <mcount>
       48 8b 47 70             mov    0x70(%rdi),%rax
       49 89 ff                mov    %rdi,%r15
```

```
<posix_cpu_timer_set>:
       e8 eb 3c 0b 00          callq  ffffffff810f0af0 <__fentry__>
       41 57                   push   %r15
       41 56                   push   %r14
       49 89 ff                mov    %rdi,%r15
       41 55                   push   %r13
       41 54                   push   %r12
       49 89 d5                mov    %rdx,%r13
       55                      push   %rbp
       53                      push   %rbx
```

fentry

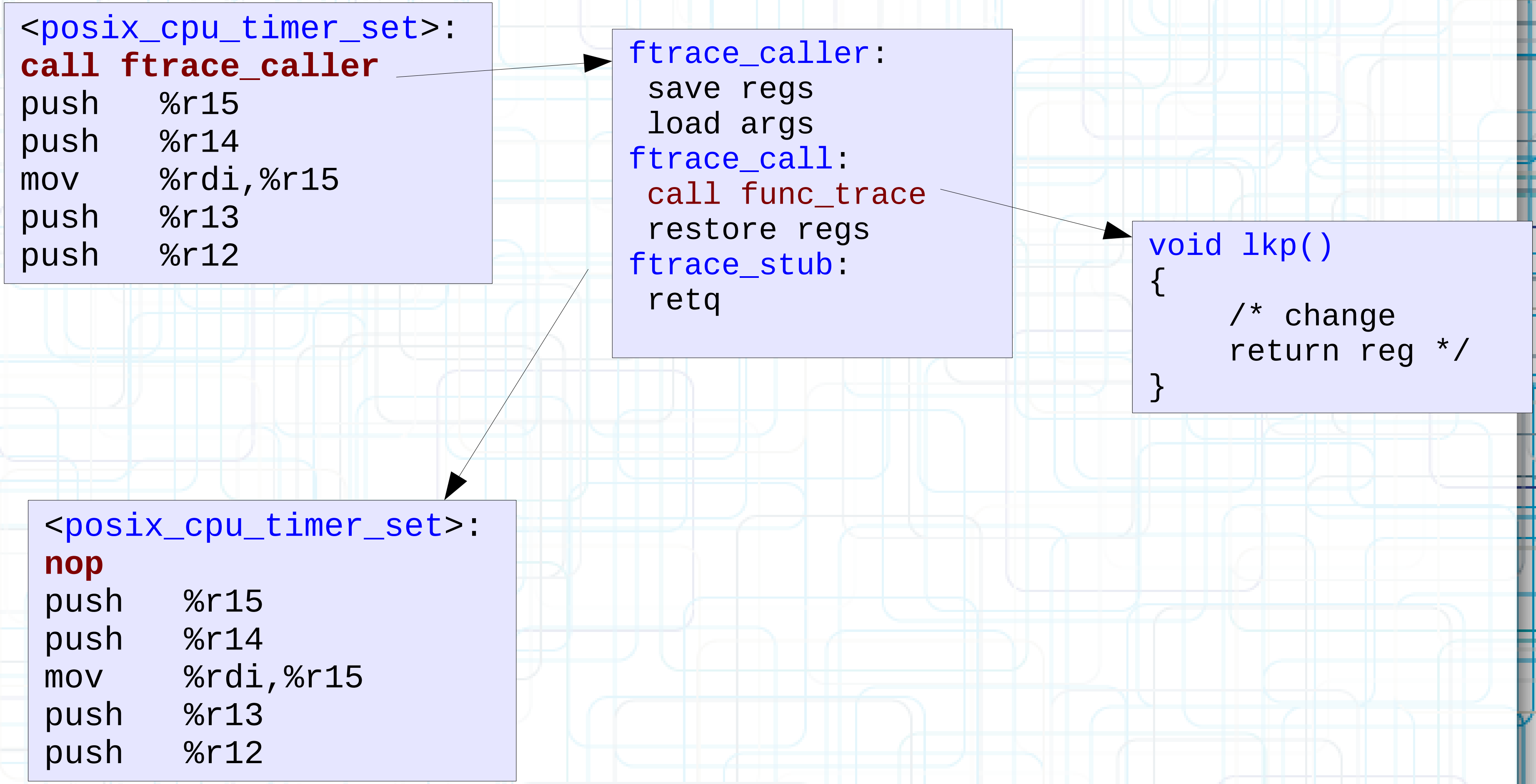


```
<posix_cpu_timer_set>:
call ftrace_caller
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call func_trace
 restore regs
ftrace_stub:
 retq
```

```
void func_trace()
{
    /* trace */
}
```

Live Kernel Patching!



```
<posix_cpu_timer_set>:
call ftrace_caller
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call func_trace
 restore regs
ftrace_stub:
 retq
```

```
void lkp()
{
    /* change
    return reg */
}
```

```
<posix_cpu_timer_set>:
nop
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

Another Solution

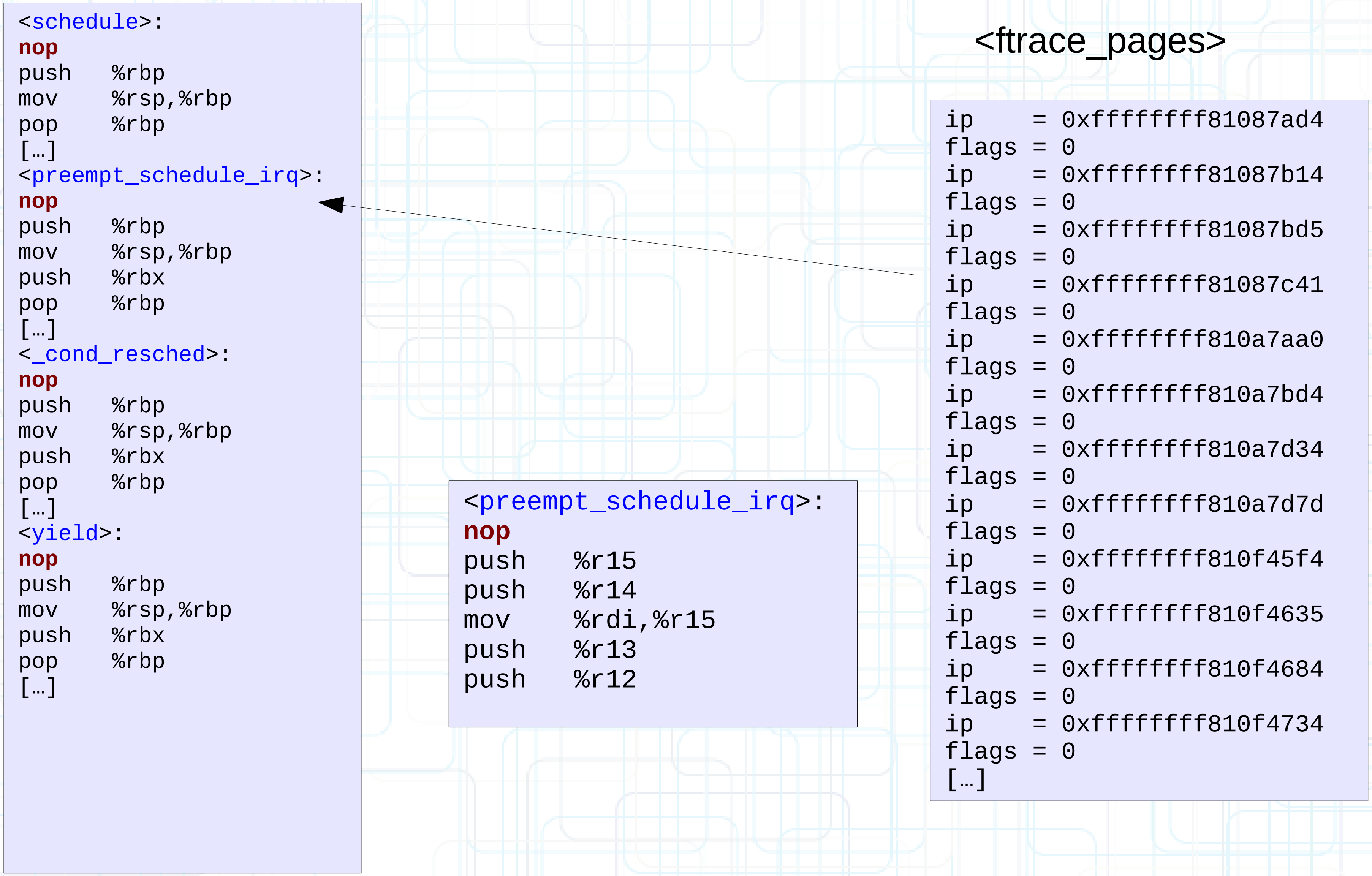
vmlinux:

```
<schedule>:
nop
push   %rbp
mov    %rsp,%rbp
pop    %rbp
[…]
<preempt_schedule_irq>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<_cond_resched>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<yield>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

Another Solution



vmlinux:

```
<schedule>:
nop
push   %rbp
mov    %rsp,%rbp
pop    %rbp
[…]
<preempt_schedule_irq>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<_cond_resched>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<yield>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
```

```
<preempt_schedule_irq>:
nop
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffff81087c41
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

Another Solution

vmlinux:

```
<schedule>:
nop
push   %rbp
mov    %rsp,%rbp
pop    %rbp
[…]
<preempt_schedule_irq>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<_cond_resched>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<yield>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffffa0014466
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

```
<preempt_schedule_irq>:
nop
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

Another Solution

vmlinux:

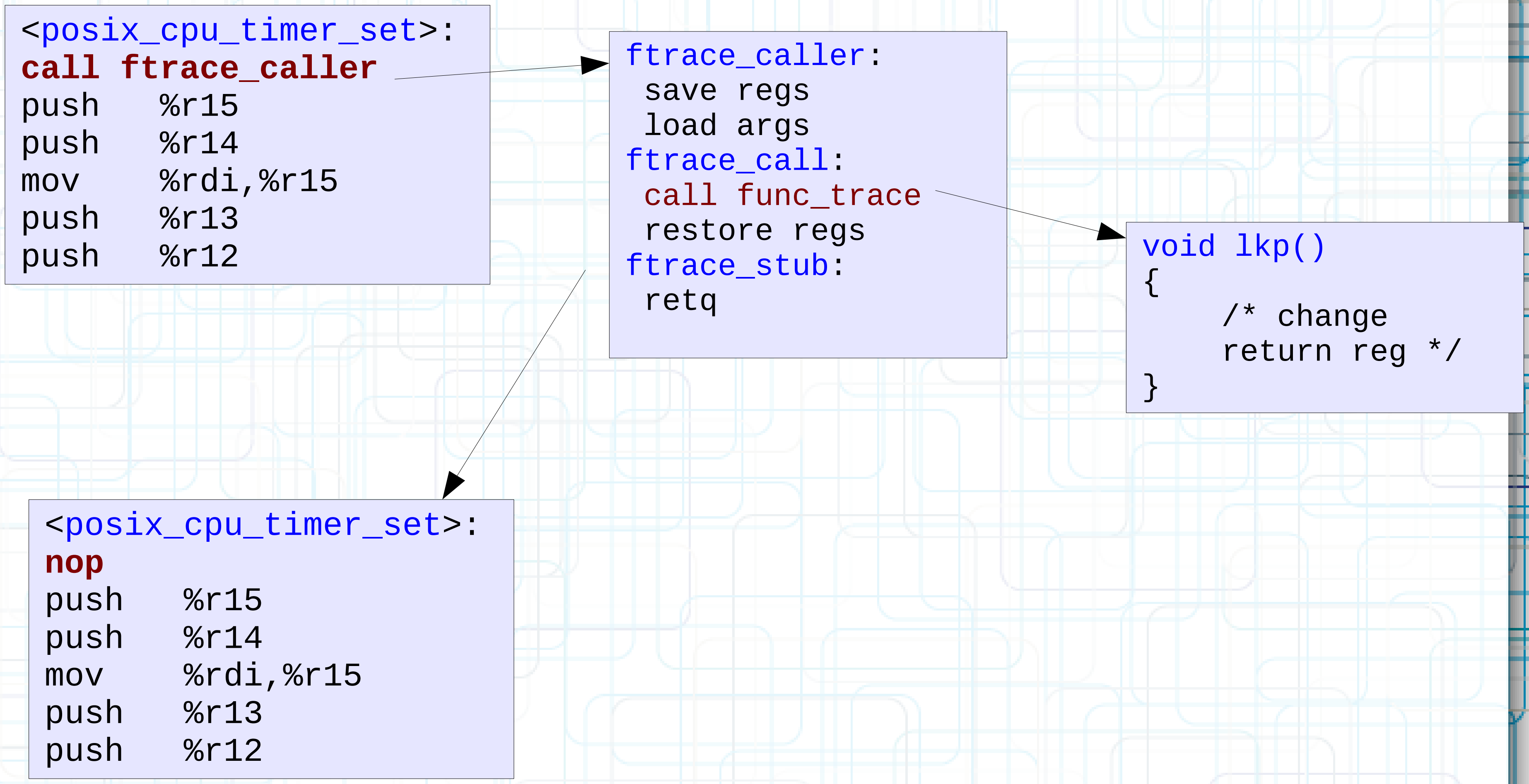
```
<schedule>:
nop
push   %rbp
mov    %rsp,%rbp
pop    %rbp
[…]
<preempt_schedule_irq>:
jmp preempt_sched2
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<_cond_resched>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
<yield>:
nop
push   %rbp
mov    %rsp,%rbp
push   %rbx
pop    %rbp
[…]
```

```
<preempt_schedule_irq>:
nop
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

<ftrace_pages>

```
ip    = 0xffffffff81087ad4
flags = 0
ip    = 0xffffffff81087b14
flags = 0
ip    = 0xffffffff81087bd5
flags = 0
ip    = 0xffffffffa0014466
flags = 0
ip    = 0xffffffff810a7aa0
flags = 0
ip    = 0xffffffff810a7bd4
flags = 0
ip    = 0xffffffff810a7d34
flags = 0
ip    = 0xffffffff810a7d7d
flags = 0
ip    = 0xffffffff810f45f4
flags = 0
ip    = 0xffffffff810f4635
flags = 0
ip    = 0xffffffff810f4684
flags = 0
ip    = 0xffffffff810f4734
flags = 0
[…]
```

Instead of this



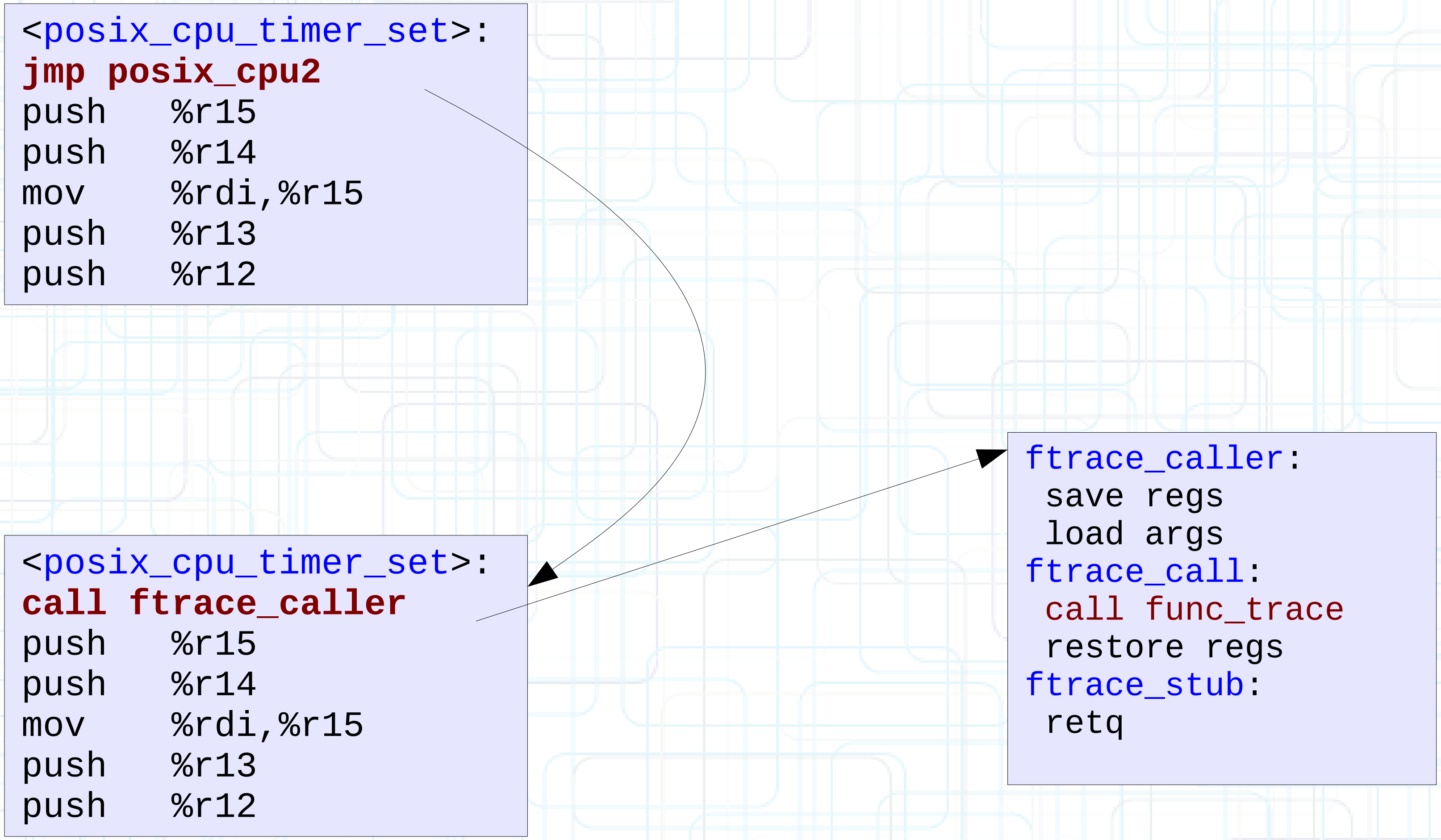
```
<posix_cpu_timer_set>:
call ftrace_caller
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call func_trace
 restore regs
ftrace_stub:
 retq
```

```
void lkp()
{
    /* change
    return reg */
}
```

```
<posix_cpu_timer_set>:
nop
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

Have this!



```
<posix_cpu_timer_set>:
jmp posix_cpu2
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

```
<posix_cpu_timer_set>:
call ftrace_caller
push   %r15
push   %r14
mov    %rdi,%r15
push   %r13
push   %r12
```

```
ftrace_caller:
 save regs
 load args
ftrace_call:
 call func_trace
 restore regs
ftrace_stub:
 retq
```

Questions?

Yeah right!
Like we have time