faster uprobes

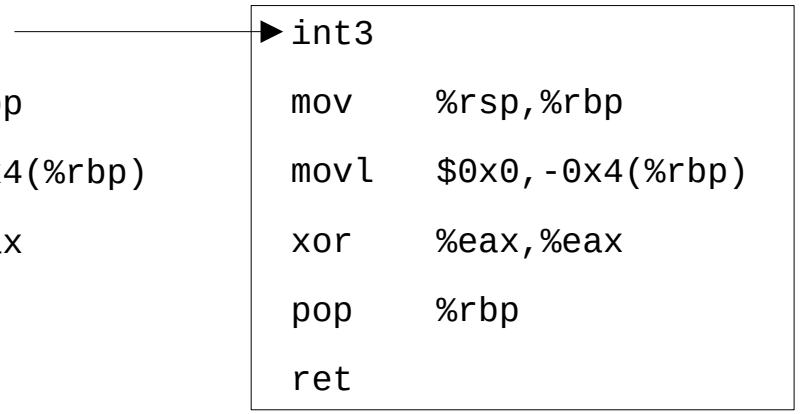
jiri olsa / isovalent at cisco

UPROBE

- **user space probe**
- **implements USDT**
- **x86 specific**

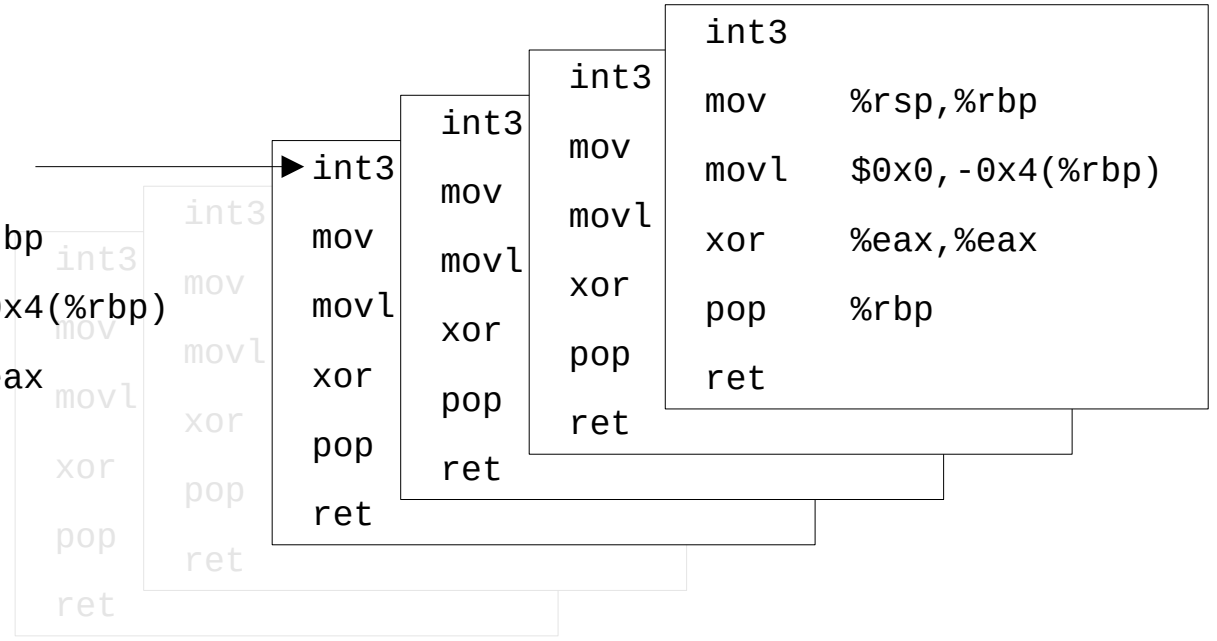
INSTALL

```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```



INSTALL

```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```



BREAKPOINT TRAP

```
int3
mov    %rsp,%rbp
movl   $0x0,-0x4(%rbp)
xor    %eax,%eax
pop    %rbp
ret
```

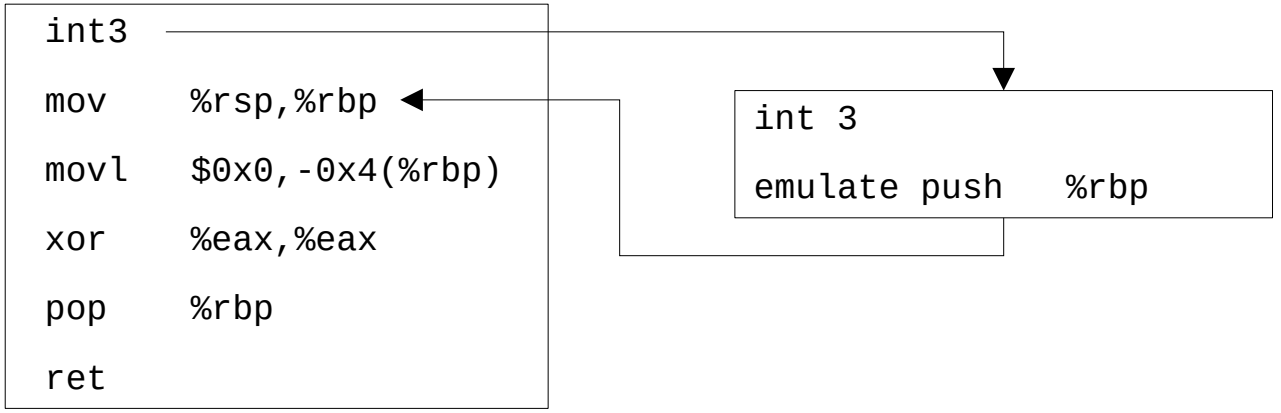
execute handlers

execute original instruction

single step or emulation

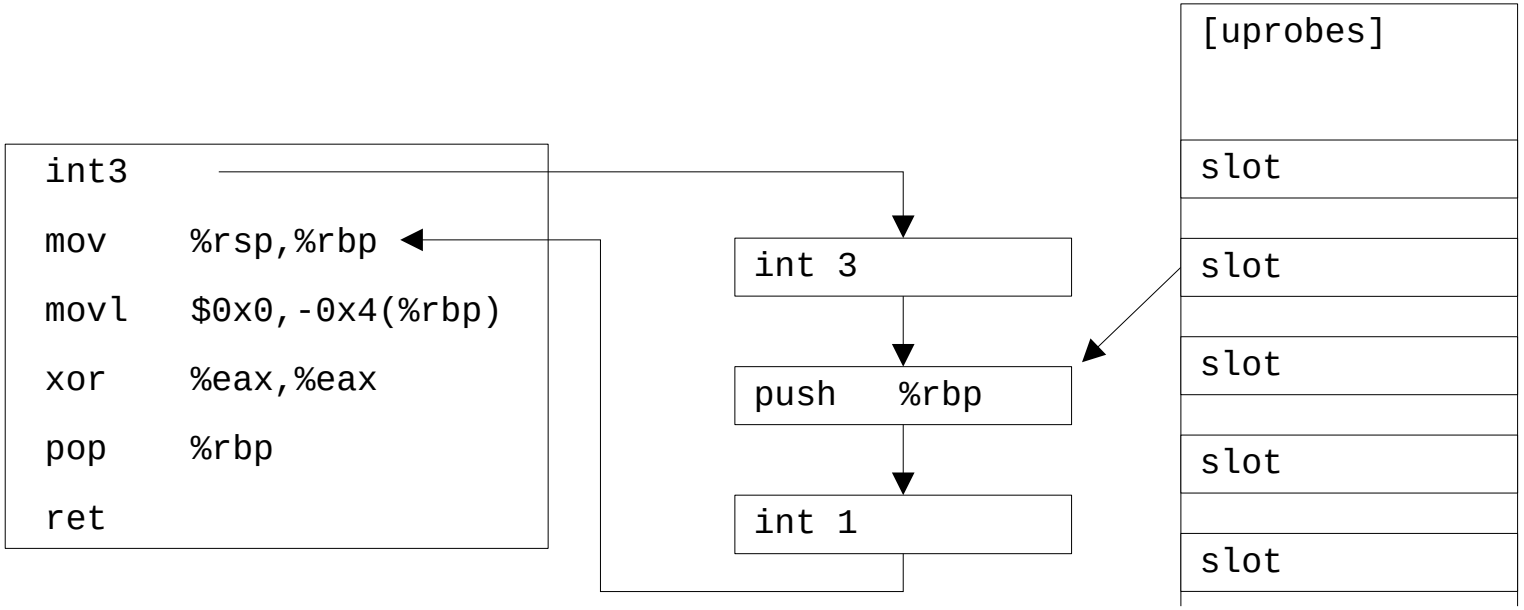
EMULATION

- **certain instructions can be emulated**
- **skips the second trap**
- **push/jmp/call/nop**



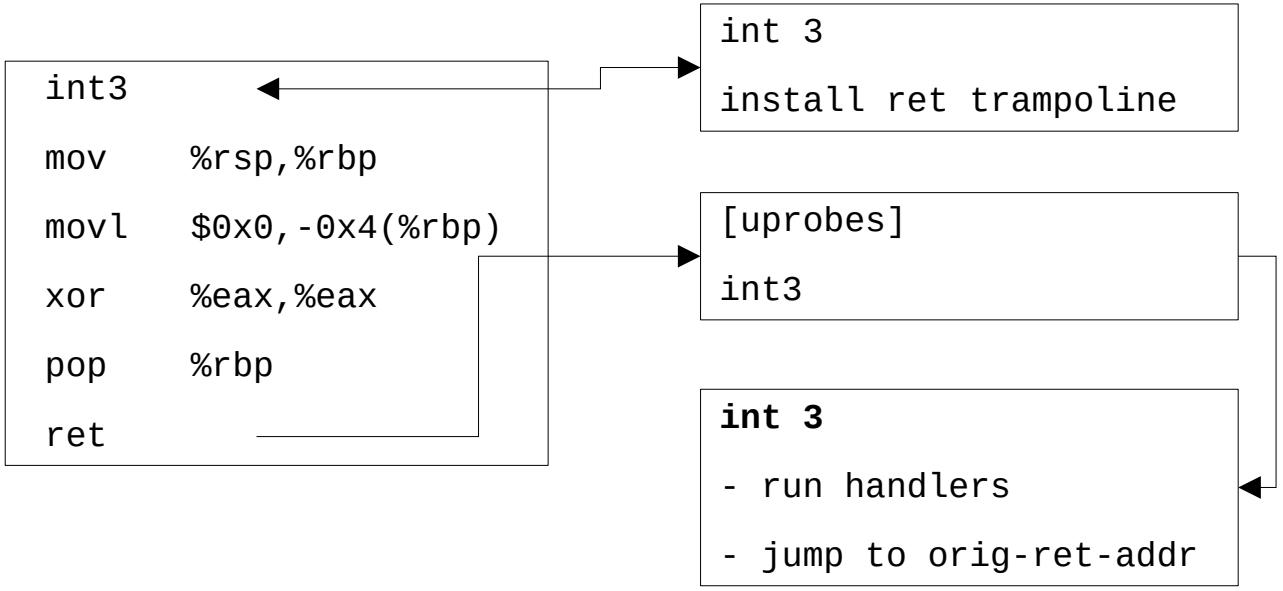
SINGLE STEP

- **copy origin instruction in XOL area**
- **setup single step**



RETURN UPROBE

- **redirect return address on user stack**
- **assumes entry probe is on function entry**



BENCH

- **trigger bench**
- **nop/push/ret**

```
# ./tools/testing/selftests/bpf/benchs/run_bench_uprobes.sh
usermode-count :  227.709 ± 1.858M/s
syscall-count  :    2.113 ± 0.047M/s
uprobe-nop     :    0.409 ± 0.005M/s
uprobe-push    :    0.379 ± 0.001M/s
uprobe-ret     :    0.185 ± 0.009M/s
uretprobe-nop  :    0.085 ± 0.000M/s
uretprobe-push :    0.094 ± 0.001M/s
uretprobe-ret  :    0.076 ± 0.000M/s
```

RECENT FIXES

- **uprobe speedups [Andrii]**

```
uprobe-nop     :    2.878 ± 0.017M/s (+5.5%, total +8.3%)
uprobe-push    :    2.753 ± 0.013M/s (+5.3%, total +10.2%)
uprobe-ret     :    1.142 ± 0.010M/s (+3.8%, total +3.8%)
uretprobe-nop  :    1.444 ± 0.008M/s (+3.5%, total +6.5%)
uretprobe-push :    1.410 ± 0.010M/s (+4.8%, total +7.1%)
uretprobe-ret  :    0.816 ± 0.002M/s (+2.0%, total +3.9%)
```

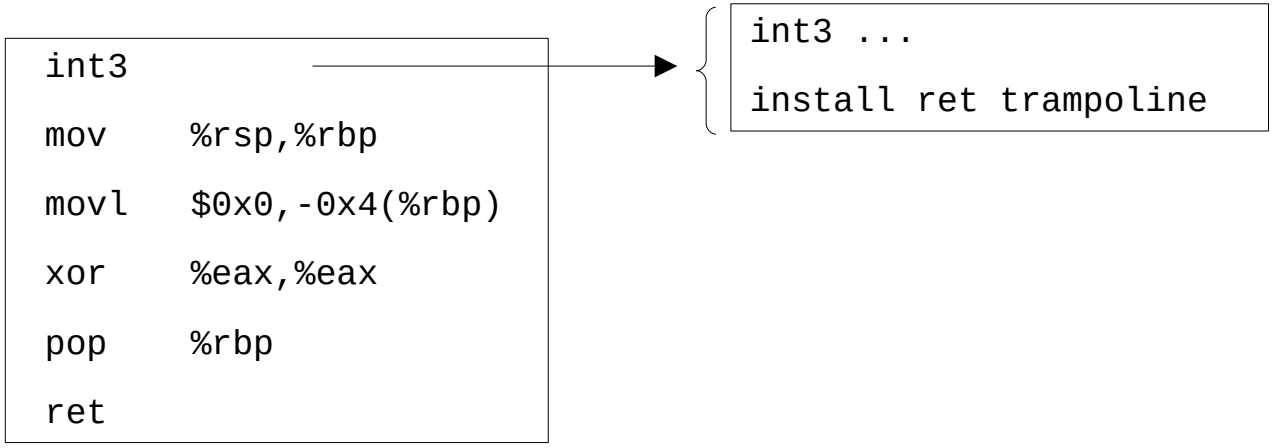
RECENT FIXES

- **reduce contention [Jonathan Haslam]**

```
… Improvements are in the order of 22 - 68% with this particular
Benchmark (mean = 43%).
```

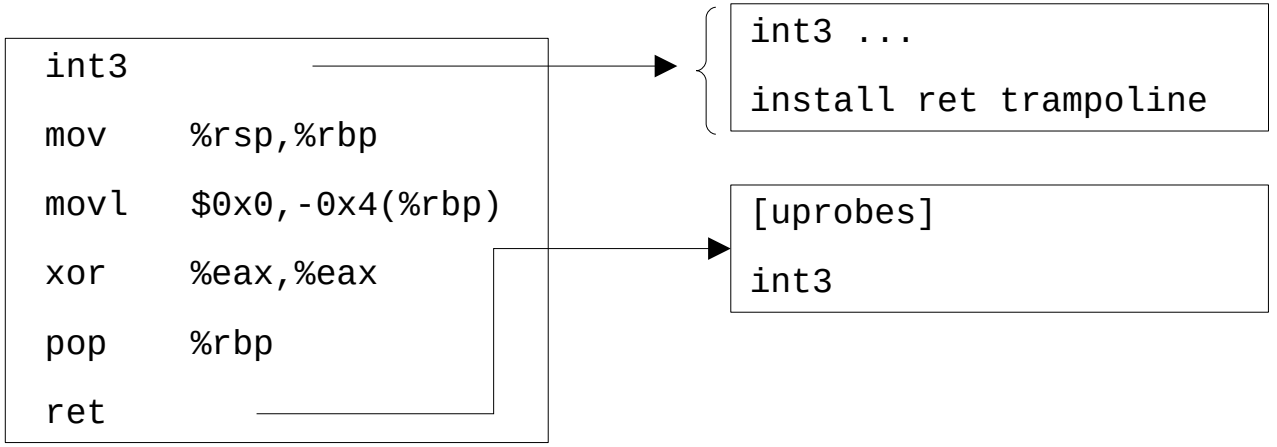
URETPROBE SPEEDUP

- **replace int3 with new uretprobe syscall**
- **faster on x86**



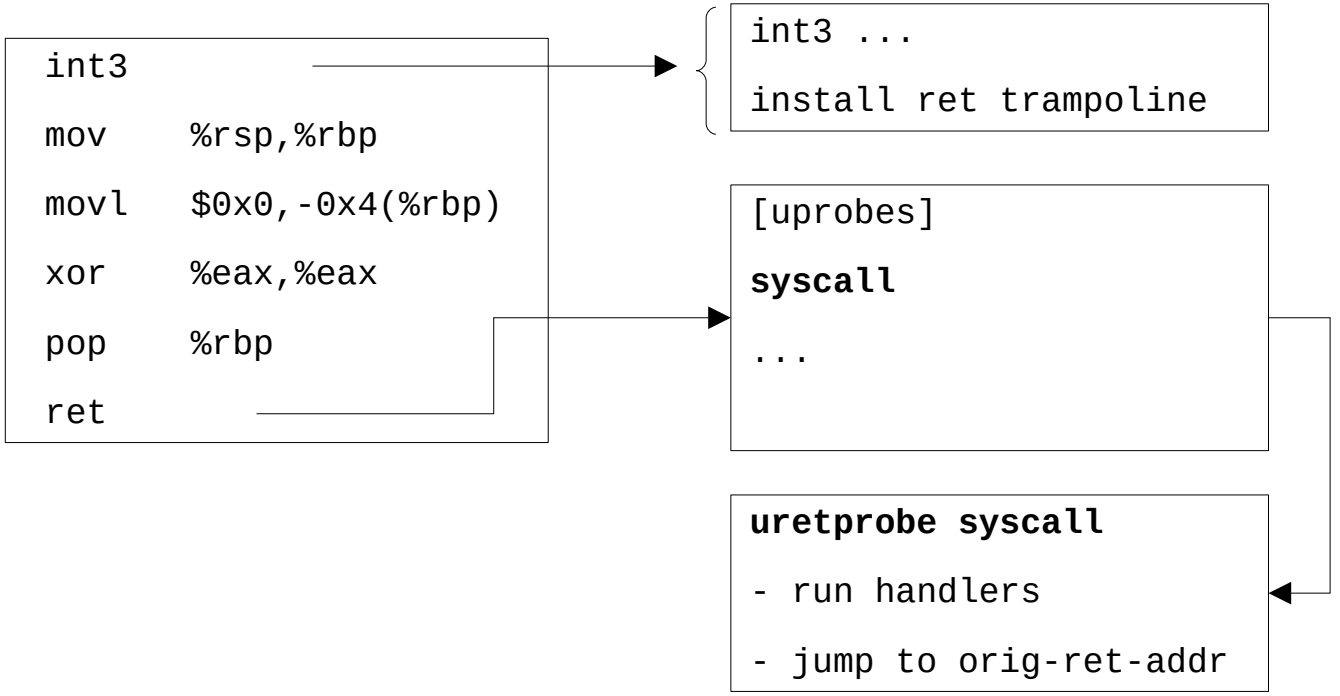
URETPROBE SPEEDUP

- **replace int3 with uretprobe syscall**
- **much (3x) faster on x86**



URETPROBE SPEEDUP

- **replace int3 with uretprobe syscall**
- **much (3x) faster on x86**



URETPROBE SPEEDUP

Intel 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz

```
uretprobe-nop  :    1.969 ± 0.002M/s  < 31% speed up
uretprobe-push :    1.910 ± 0.000M/s  < 31% speed up
uretprobe-ret  :    0.934 ± 0.000M/s  < 14% speed up
```

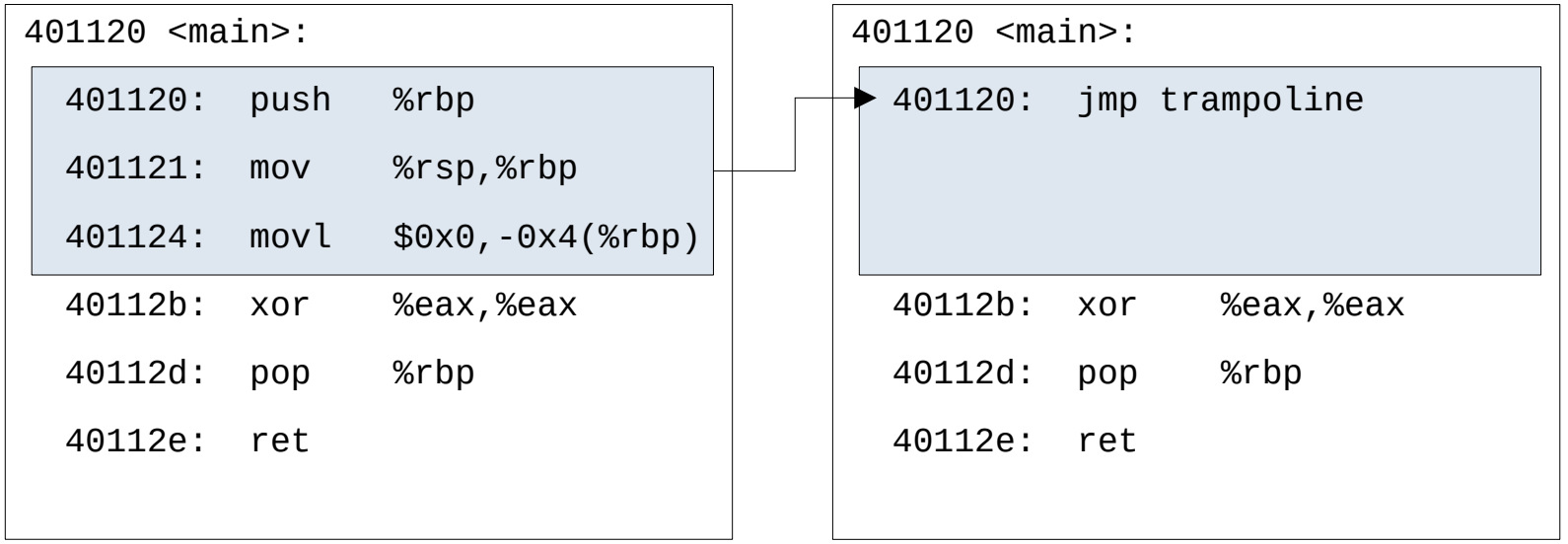
AMD Ryzen 7 5700U

```
uretprobe-nop  :    0.860 ± 0.001M/s  < 10% speed up
uretprobe-push :    0.818 ± 0.001M/s  < 10% speed up
uretprobe-ret  :    0.578 ± 0.000M/s  <  7% speed up
```

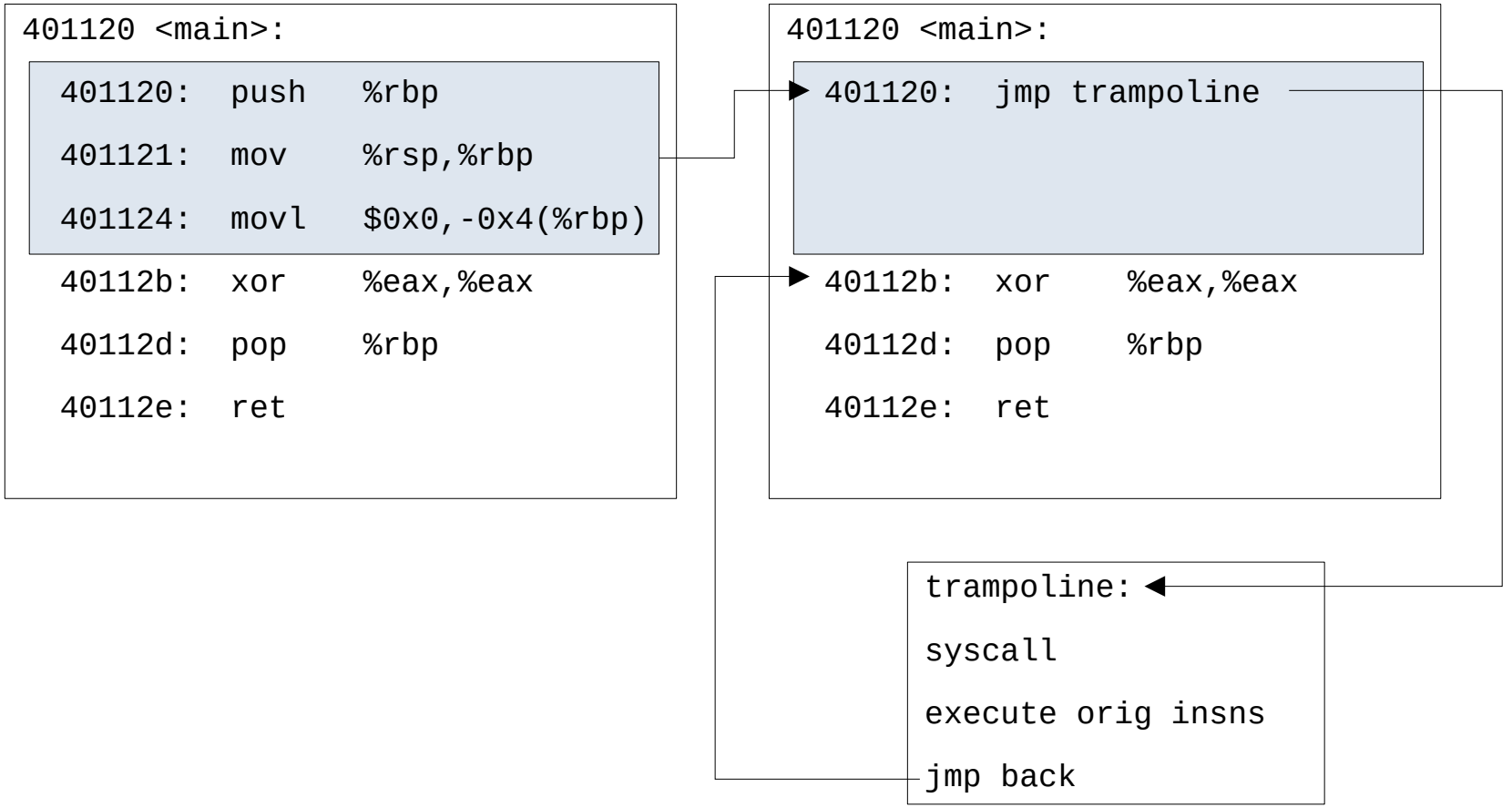
UPROBE SPEEDUP

```
401120 <main>:
  401120:  push   %rbp
  401121:  mov    %rsp,%rbp
  401124:  movl   $0x0,-0x4(%rbp)
  40112b:  xor    %eax,%eax
  40112d:  pop    %rbp
  40112e:  ret
```

UPROBE SPEEDUP



UPROBE SPEEDUP

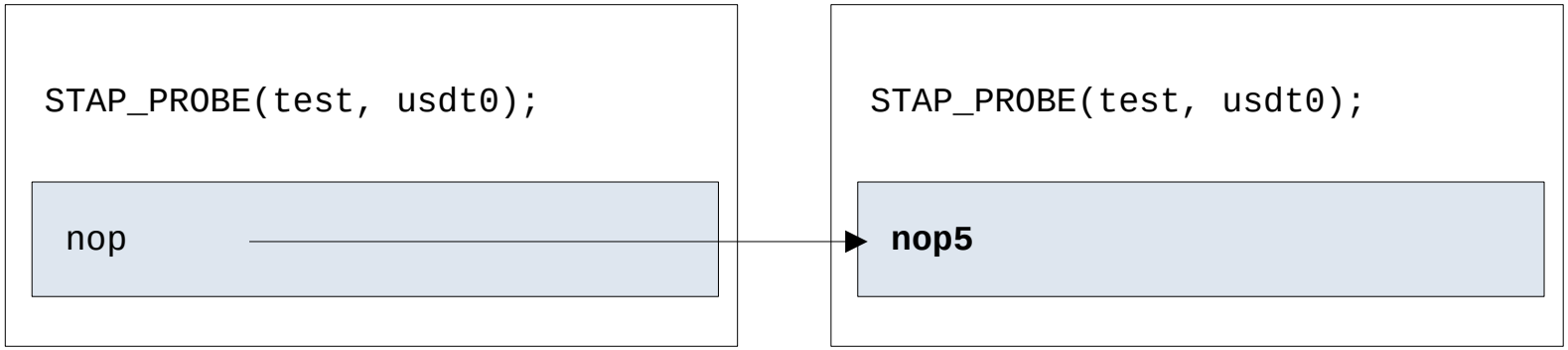


USDT SPEEDUP

```
STAP_PROBE(test, usdt0);
```

```
nop
```

USDT SPEEDUP

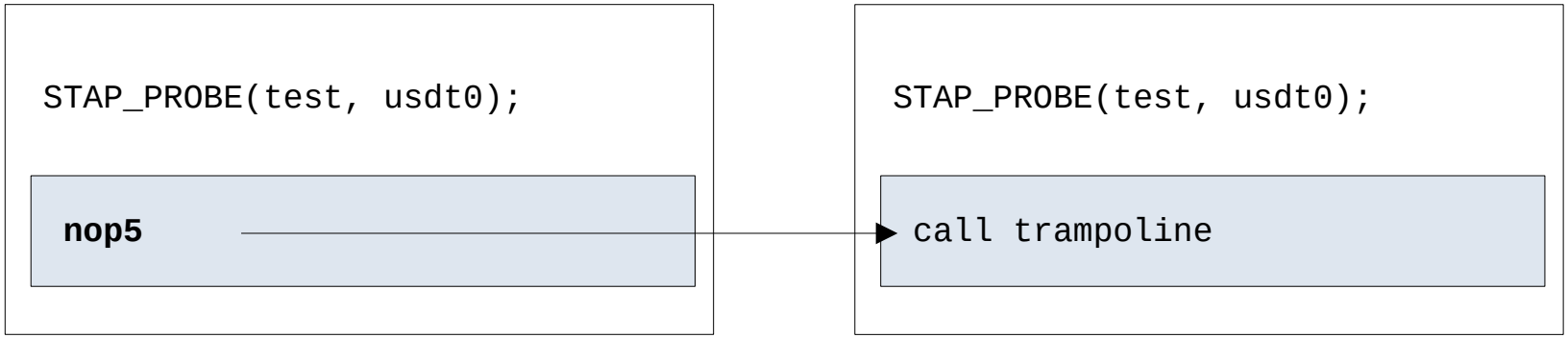


USDT SPEEDUP

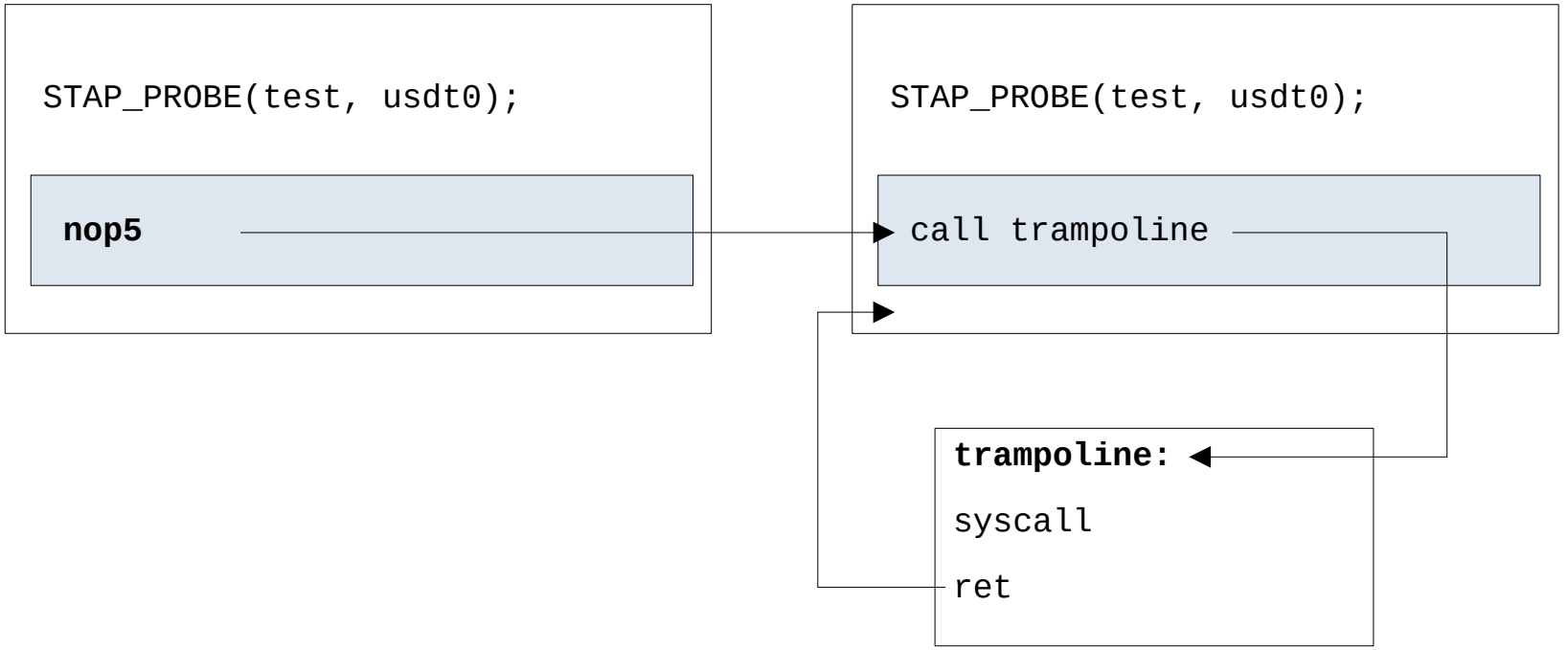
```
STAP_PROBE(test, usdt0);
```

nop5

USDT SPEEDUP



USDT SPEEDUP



ISSUES

- **safe 5 bytes update**
- **original instructions execution**
- **multiple userspace trampolines**
- **...**

USDT SPEEDUP

Intel 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz

```
usermode-count: Summary: hits  233.854 ± 0.470M/s  ...
base:           Summary: hits    3.290 ± 0.005M/s  ...
fix:            Summary: hits    7.930 ± 0.111M/s  ... 2.5x speed up
```

SAFE UPDATE

- use the text_poke_bp way
  - write int3
  - write jmp/call offset
  - write jmp/call opcode
- int 3 handler emulates the jump while it’s updated

USER SPACE TRAMPOLINES

- **can’t access whole userspace with 4 bytes offset**
- **trampoline close to uprobe**
- **5 bytes jump? alternatives?**

thanks, questions..