Valgrind

register allocator overhaul

Ivo Raisr

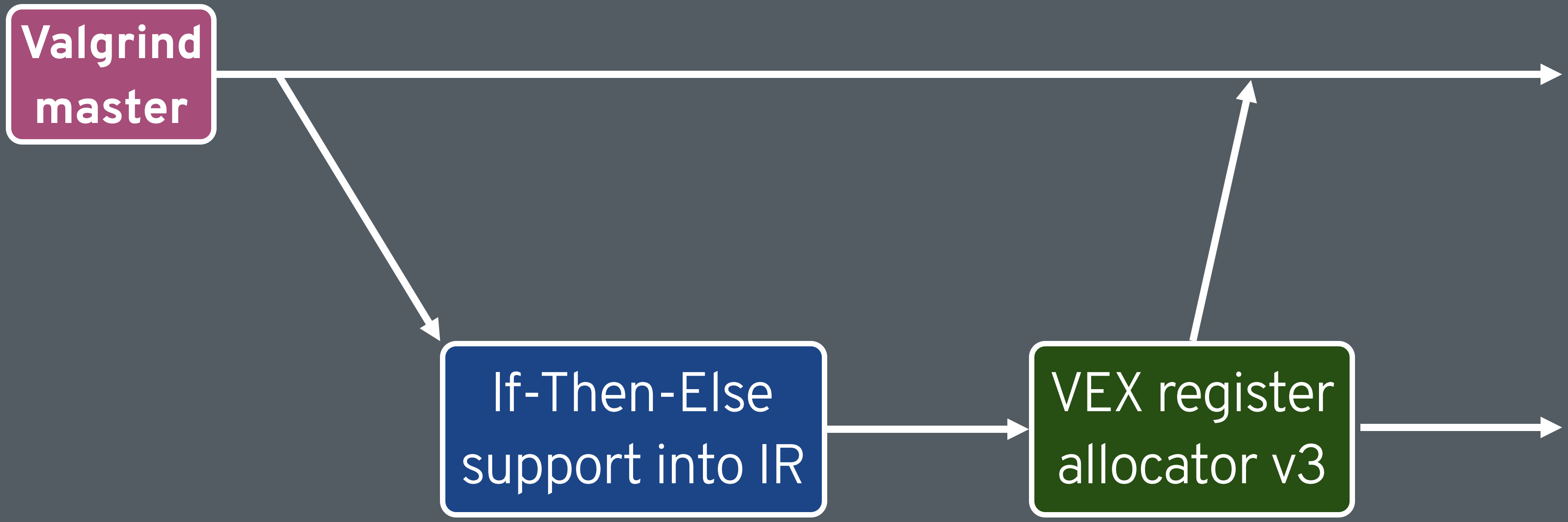
FOSDEM 2018

Ivo Raisr

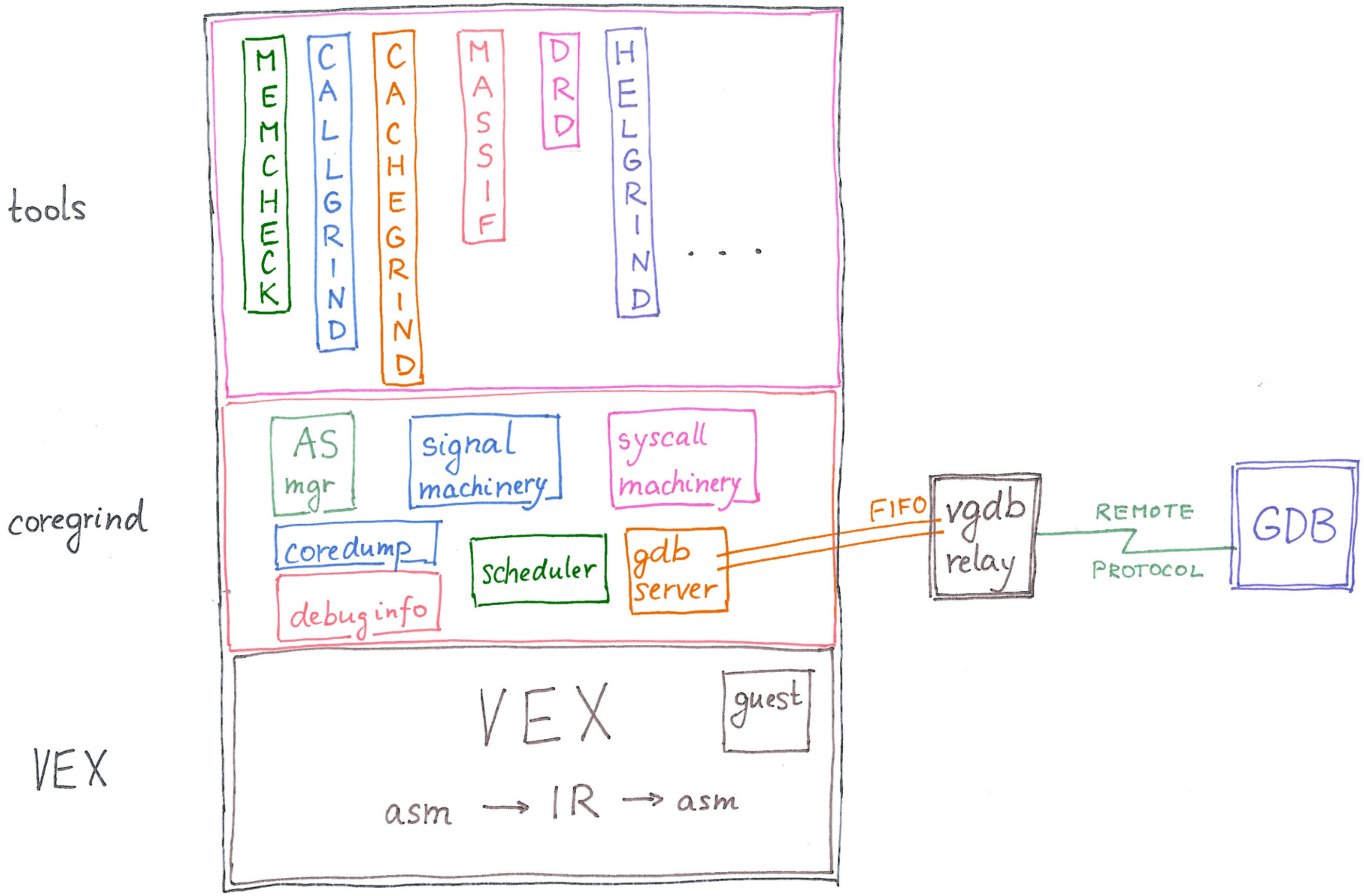
39.6

GNU Toolchain

Why?


Valgrind master
If-Then-Else support into IR
VEX register allocator v3

VALGRIND ARCHITECTURE

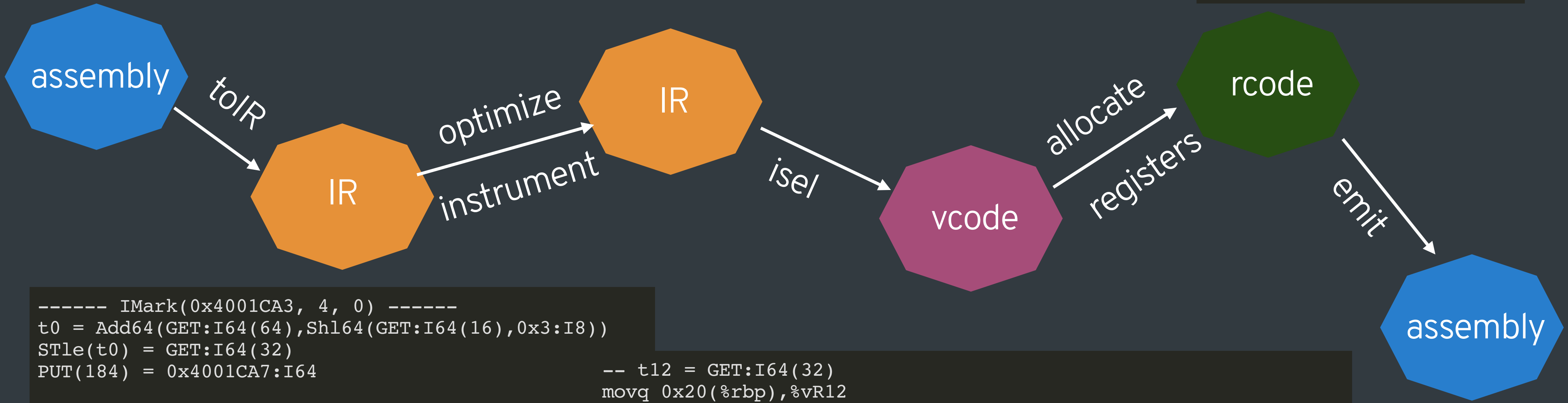


VEX operation

```
------ IMark(0x4001CA3, 4, 0) ------
t12 = GET:I64(32)
STle(Add64(GET:I64(64),Shl64(GET:I64(16),0x3:I8))) = t12
```

```
0x4001CA3:  movq %rdx,(%rs
```

```
movq 0x20(%rbp),%r10
movq 0x40(%rbp),%r9
movq 0x10(%rbp),%r8
movq %r10,0x0(%r9,%r8,8)
```



```
------ IMark(0x4001CA3, 4, 0) ------
t0 = Add64(GET:I64(64),Shl64(GET:I64(16),0x3:I8))
STle(t0) = GET:I64(32)
PUT(184) = 0x4001CA7:I64
```

```
-- t12 = GET:I64(32)
movq 0x20(%rbp),%vR12

-- STle(Add64(GET:I64(64),Shl64(GET:I64(16),0x3:I8))) = t12
movq 0x40(%rbp),%vR24
movq 0x10(%rbp),%vR25
movq %vR12,0x0(%vR24,%vR25,8)
```

VEX register allocator

```
(evCheck) decl 0x8(%rbp); jns nofail; jm
movq 0x40(%rbp),%vR65
movq 0x10(%rbp),%vR66
leaq 0x0(%vR65,%vR66,8),%vR8
movq 0x3C0(%rbp),%vR35
movq 0x20(%rbp),%vR12
movq 0x3E0(%rbp),%vR67
movq 0x3B0(%rbp),%vR69
movq %vR69,%vR68
shlq $3,%vR68
movq %vR67,%vR70
orq %vR68,%vR70
callnz[0,RLPri_None] 0x58024160
movq %vR8,%rdi
movq %vR35,%rsi
call[2,RLPri_None] 0x58023660
movq %vR12,(%vR8)
movq %vR35,%vR75
notq %vR75
movq %vR12,%vR74
...
```

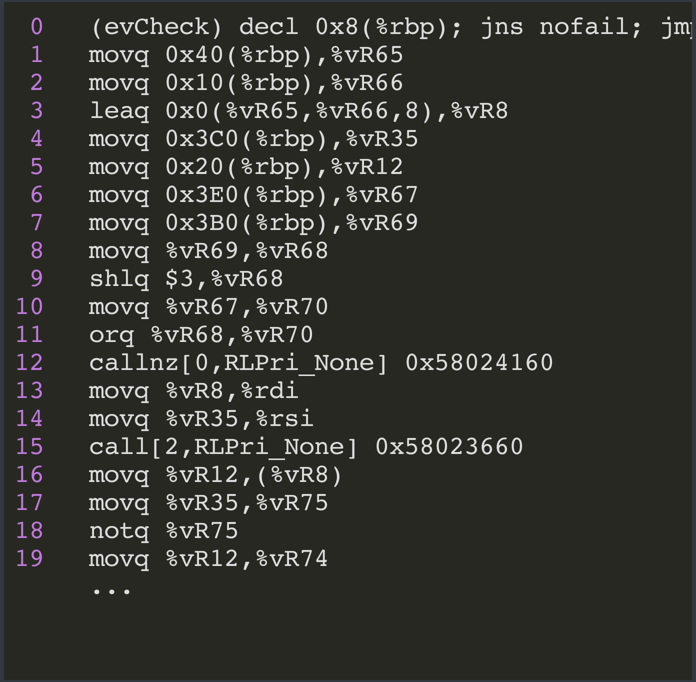
vcode

```
(evCheck) decl 0x8(%rbp); jns nofail; jmp
movq 0x40(%rbp),%r10
movq 0x10(%rbp),%r9
leaq 0x0(%r10,%r9,8),%rbx
movq 0x3C0(%rbp),%r15
movq 0x20(%rbp),%r14
movq 0x3E0(%rbp),%r10
movq 0x3B0(%rbp),%r9
shlq $3,%r9
orq %r9,%r10
callnz[0,RLPri_None] 0x58024160
movq %rbx,%rdi
movq %r15,%rsi
call[2,RLPri_None] 0x58023660
movq %r14,(%rbx)
movq %r15,%r10
notq %r10
movq %r14,%r9
...
```

rcode

RegAlloc Terminology

```
1   movq 0x40(%rbp), %vR65
2   movq 0x10(%rbp), %vR66
            ...
8   movq %vR69, %vR68
9   shlq $3, %vR68
10  movq %vR67, %vR70
11  orq %vR68, %vR70
12  callnz[0, RLPri_None] <addr>
13  movq %vR8, %rdi
14  movq %vR35, %rsi
15  call[2, RLPri_None] <addr>
            ...
```



```
0    (evCheck) decl 0x8(%rbp); jns nofail; jm
1    movq 0x40(%rbp),%vR65
2    movq 0x10(%rbp),%vR66
3    leaq 0x0(%vR65,%vR66,8),%vR8
4    movq 0x3C0(%rbp),%vR35
5    movq 0x20(%rbp),%vR12
6    movq 0x3E0(%rbp),%vR67
7    movq 0x3B0(%rbp),%vR69
8    movq %vR69,%vR68
9    shlq $3,%vR68
10   movq %vR67,%vR70
11   orq %vR68,%vR70
12   callnz[0,RLPri_None] 0x58024160
13   movq %vR8,%rdi
14   movq %vR35,%rsi
15   call[2,RLPri_None] 0x58023660
16   movq %vR12,(%vR8)
17   movq %vR35,%vR75
18   notq %vR75
19   movq %vR12,%vR74
     ...
```

vcode

RegAlloc v3 Passes

```
         ...
 8   movq %vR69, %vR68
 9   shlq $3, %vR68
10   movq %vR67, %vR70
11   orq %vR68, %vR70
12   callnz[0, RLPri_None] <addr>
13   movq %vR8, %rdi
14   movq %vR35, %rsi
15   call[2, RLPri_None] <addr>
         ...
21   movq %vR70, %vR9
```

1. scan insns

   %vR69 %rdi

2. coalescing

   %vR67 -> %vR70 -> %vR9

3. spill slots

4. process insns

   ```
   %vR68 ... %rdi
   %vR69 ... %rax
   %vR70 ... %r9
   ```

RegAlloc v3 State

vreg state

%vR68	… [8, 12)	… %rdx	… [12]
%vR69	… [7, 9)	… ---	… [10]
%vR70	… [10, 12)	… %r9	… [5]

live after
dead before
real reg
spill slot

```
…
8   movq %vR69, %vR68
9   shlq $3, %vR68
10  movq %vR67, %vR70
11  orq %vR68, %vR70
12  callnz[0, RLPri_None] <addr>
13  movq %vR8, %rdi
14  movq %vR35, %rsi
15  call[2, RLPri_None] <addr>
…
21  movq %vR70, %vR9
```

%vR67 -> %vR70 -> %vR9

RegAlloc v3 State II.

rreg state

```
%rdx  ... %vR68
%rcx  ... ---
%rdi  ... [reserved]
```

rreg universe

```
%r12, %r13, %r14, %r15, %rbx,
%rsi, %rdi, %r8, %r9, %r10
```

HRcInt64

```
        ...
8   movq %vR69, %vR68
9   shlq $3, %vR68
10  movq %vR67, %vR70
11  orq %vR68, %vR70
12  callnz[0, RLPri_None] <addr>
13  movq %vR8, %rdi
14  movq %vR35, %rsi
15  call[2, RLPri_None] <addr>
        ...
21  movq %vR70, %vR9
```

Processing insn (simple cases)

```
movq 0x40(%rbp), %vR68
        ↓
movq 0x40(%rbp), %r10
```

vreg state		rreg state	
%vR68	… %r10	%r9	… ---
%vR70	… ---	%r10	… %vR68

```
orq %vR68, %vR70
        ↓
orq %r10, %r9
```

vreg state		rreg state	
%vR68	… %r10	%r9	… %vR70
%vR70	… %r9	%r10	… %vR68

```
 movq %v70, %rsi
call[2, RLPri_None] <addr>
        ↓
 movq %r9, %rsi
```

vreg state		rreg state	
		%rsi	… reserved
%vR68	… %r10	%r9	… %vR70
%vR70	… %r9	%r10	… %vR68

Processing insn (spill)

```
movq 0x40(%rbp), %vR15
```

all rregs are taken,
what to do?

↓

spill slot

```
movq %r9, 0xC0A(%rbp)
movq 0x40(%rbp), %r9
```

%vR15	...	---
%vR68	...	%r10
%vR70	...	%r9

%r9	...	%vR70
%r10	...	%vR68
	...	

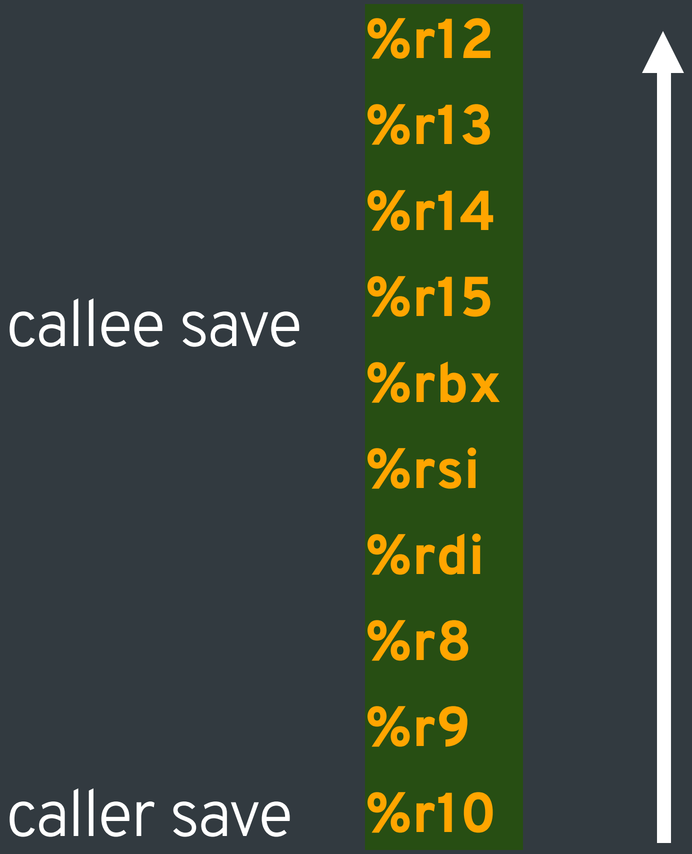
(all assigned)

Optimizations

1. **MOV vregs coalescing**
2. **reusing spill slots**
3. **vreg spilling criteria**
4. **avoid spilling if rreg == spill slot**
5. **rreg allocation strategy**
6. **direct reload**

5. rreg allocation strategy

amd64 rreg universe for HRcInt64



6. direct reload from a spill slot

```
addq %vR68, $0x9823, %vR15
```

```
%vR68  ... spilled
```

standard way

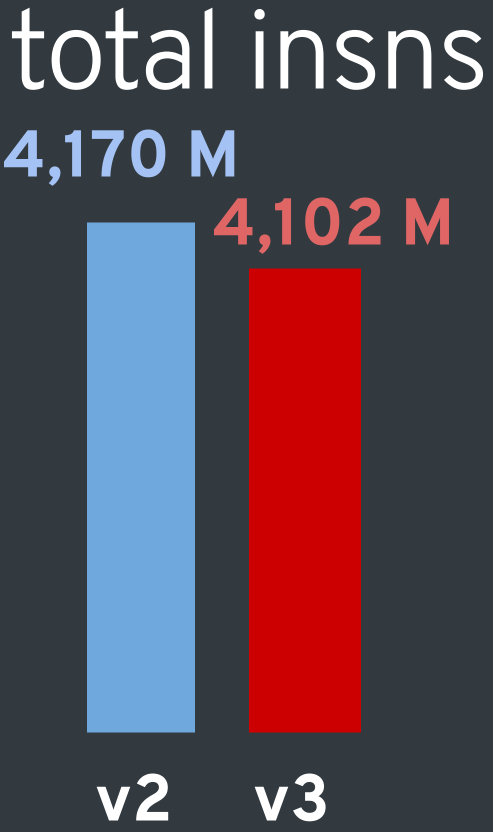
```
movq 0xC0A(%rbp), %r9
addq %r9, 0x9823, %r10
```

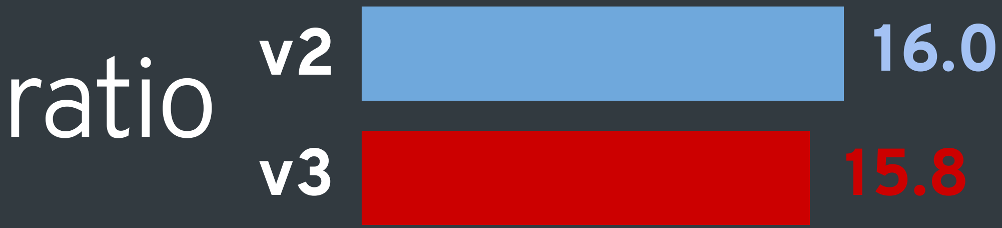
direct reload

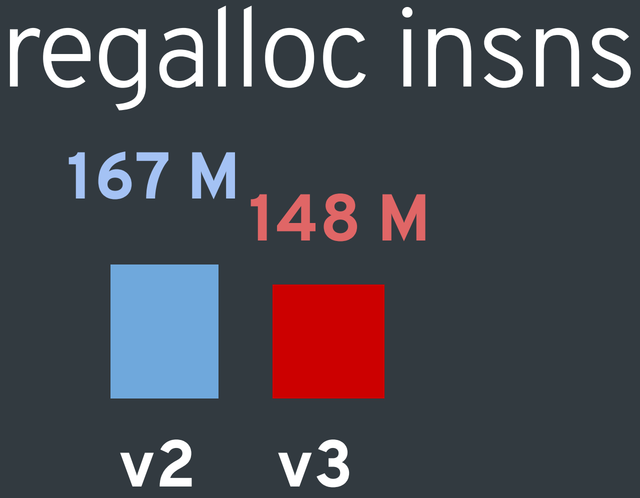
```
addq 0xC0A(%rbp), $0x9823, %r10
```

Benchmarks

Memcheck on perf/bz2, amd64







VEX register allocator v3 is now the default.

The old implementation available with:

```
--vex-regalloc-version=2
```