

ORC experiments

Wim Taymans

Principal Software Engineer

October 9, 2015

ORC... but what?

- Language and tools for describing and executing low-level computations on modern CPUs

ORC... the language

```
.function test_orc_muladd
.dest 4 d1 guint32
.source 4 s1 guint16
.source 4 s2 guint16
.temp 8 q1
.temp 8 q2
.temp 4 l1
.temp 4 l2

x2 convswl q1, s1
x2 convswl q2, s2
x2 mulll q1, q1, q2
select0ql l1, q1
select1ql l2, q1
addl d1, l1, l2
```

ORC... direct translation

```
.function test_orc_muladd
.dest 4 d1 guint32
.source 4 s1 guint16
.source 4 s2 guint16
.temp 8 q1
.temp 8 q2
.temp 4 l1
.temp 4 l2

x2 convswl q1, s1 → movq 0(%rdx), %xmm0
x2 convswl q2, s2 → movq 0(%rsi), %xmm1
x2 mulll q1, q1, q2 → pmovsxwd %xmm0, %xmm0
select0ql l1, q1 → pmovsxwd %xmm1, %xmm1
select1ql l2, q1 → pmulld %xmm1, %xmm0
addl d1, l1, l2 → movdqu %xmm0, %xmm1
                           pshufd $136, %xmm0, %xmm1
                           psrlq $32, %xmm0
                           pshufd $136, %xmm0, %xmm0
                           paddd %xmm0, %xmm1
                           movq %xmm1, 0(%rax)
```

...

ORC... or better..

...

```
.function test_orc_muladd
.dest 4 d1 guint32
.source 4 s1 guint16
.source 4 s2 guint16
.temp 8 q1
.temp 8 q2
.temp 4 l1
.temp 4 l2
```

```
x2 convswl q1, s1
x2 convswl q2, s2
x2 mulll q1, q1, q2
select0ql l1, q1
select1ql l2, q1
addl d1, l1, l2
```

```
movq 0(%rdx), %xmm0
movq 0(%rsi), %xmm1
pmaddwd %xmm1, %xmm0
movq %xmm0, 0(%rax)
```

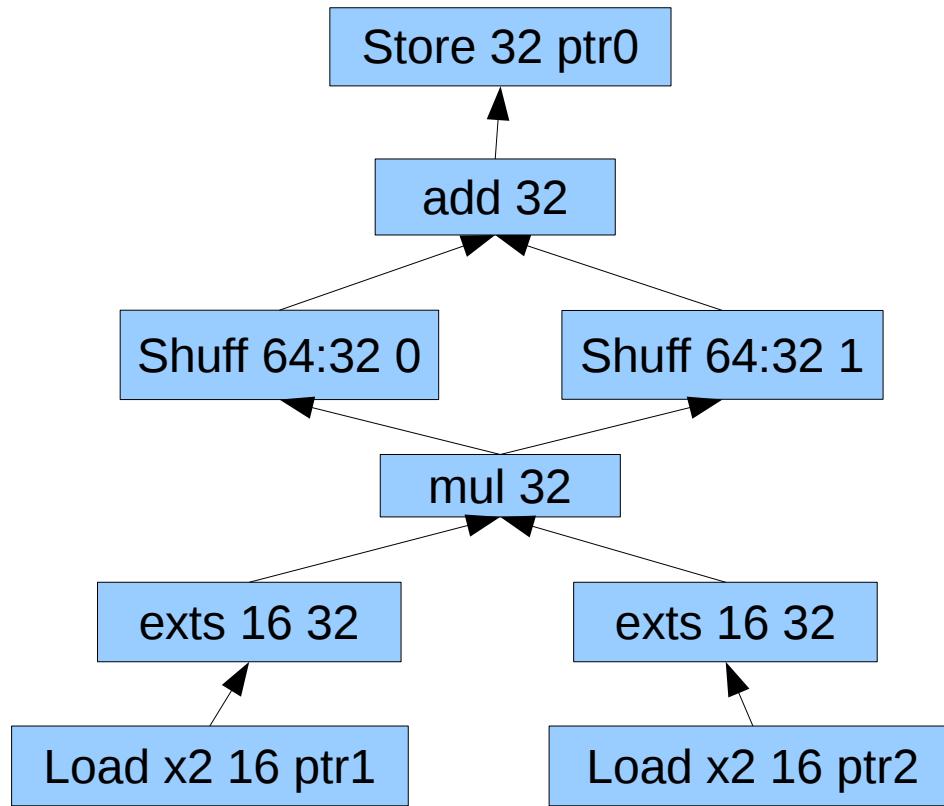
ORC... IR

```
.function test_orc_muladd
.dest 4 d1 guint32
.source 4 s1 guint16
.source 4 s2 guint16
.temp 8 q1
.temp 8 q2
.temp 4 l1
.temp 4 l2
...
t0 = load x2 16 ptr1          # x2
t1 = exts 16 t0 32
t2 = load x2 16 ptr2          # x2
t3 = exts 16 t2 32
t4 = mul 32 t1 t3             # x2
t5 = shuff 64:32 t4 t4 c0    # x1
t6 = shuff 64:32 t4 t4 c1    # x1
t7 = add 32 t5 t6             # x1
store 32 t7 ptr0              # x1

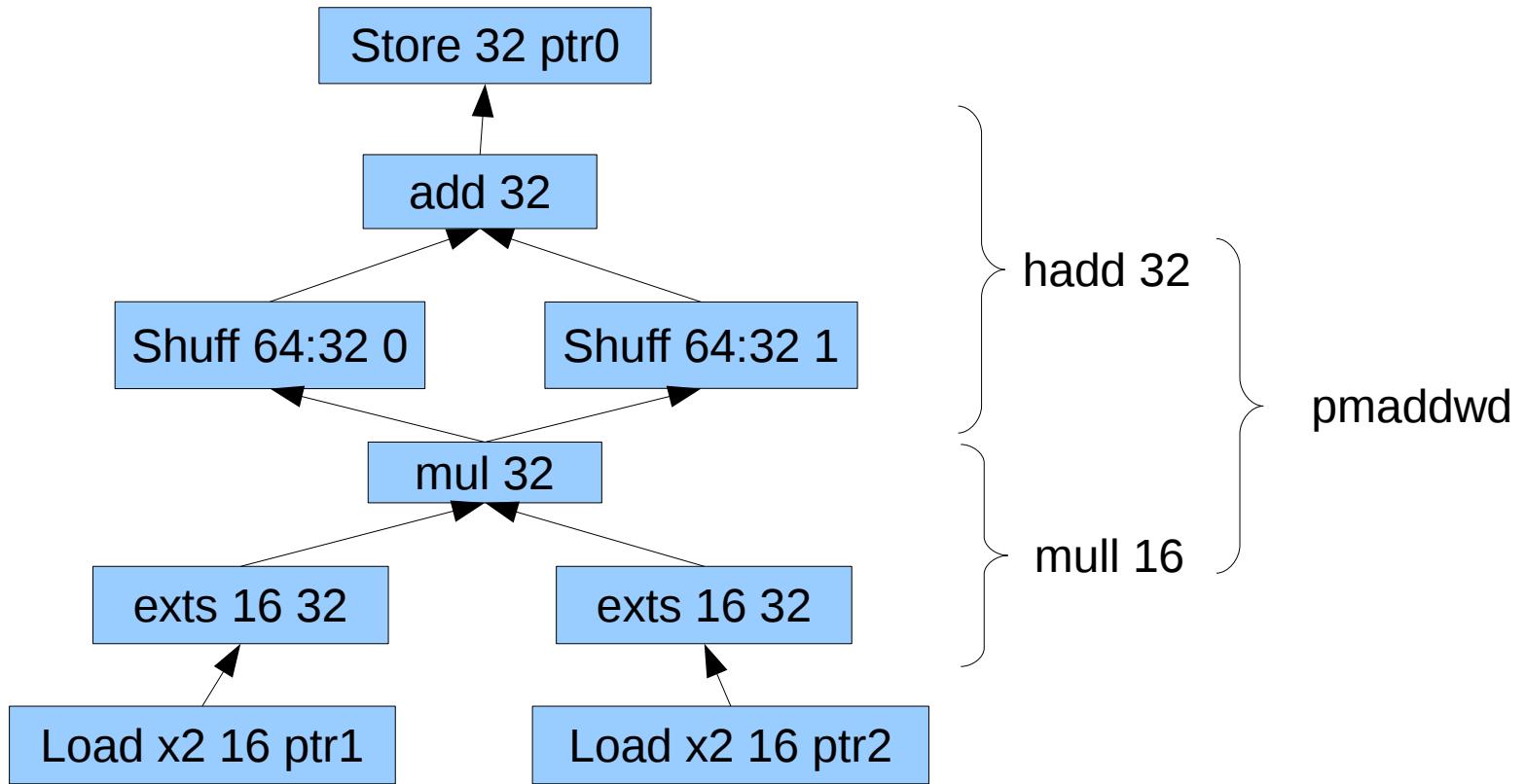
x2 convswl q1, s1
x2 convswl q2, s2
x2 mulll q1, q1, q2
select0ql l1, q1
select1ql l2, q1
addl d1, l1, l2
```

```
graph LR; x2_convswl_q1_s1[x2 convswl q1, s1] --> t0; x2_convswl_q2_s2[x2 convswl q2, s2] --> t2; x2_mulll_q1_q1_q2[x2 mulll q1, q1, q2] --> t4; select0ql_l1_q1[select0ql l1, q1] --> t5; select1ql_l2_q1[select1ql l2, q1] --> t6; adddl_d1_l1_l2[addl d1, l1, l2] --> t7; t0 = load x2 16 ptr1; t1 = exts 16 t0 32; t2 = load x2 16 ptr2; t3 = exts 16 t2 32; t4 = mul 32 t1 t3; t5 = shuff 64:32 t4 t4 c0; t6 = shuff 64:32 t4 t4 c1; t7 = add 32 t5 t6; store 32 t7 ptr0;
```

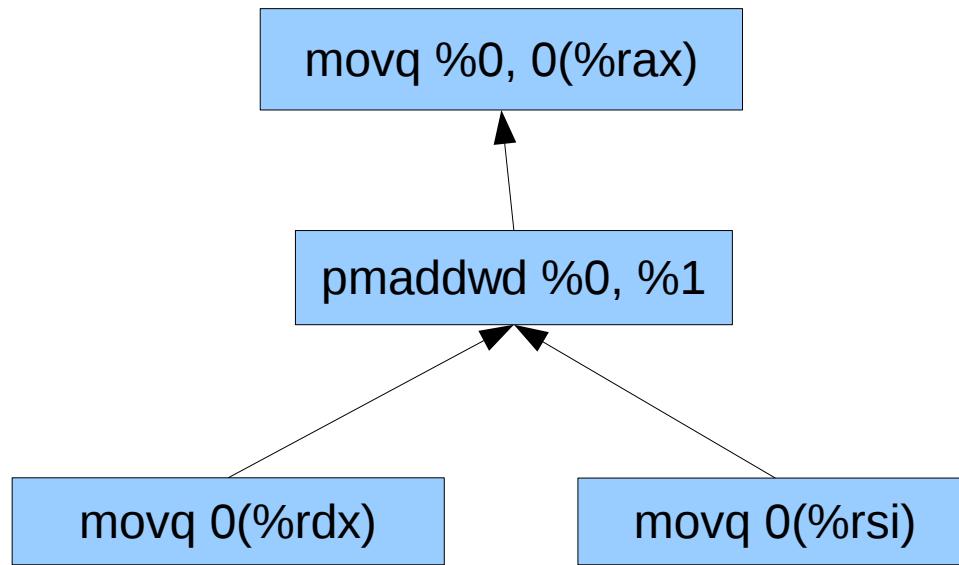
ORC... IR as tree



ORC... instruction selection



ORC... machine instruction tree



ORC... status

- Nice experiment
 - C backend compiles backup code 30% faster
 - x86 prototype backend generates much better code
- It's a lot of work
- Really would like to reuse (and improve) something existing...
 - LLVM
 - Libgccjit



<http://cgit.freedesktop.org/~wtay/orc/log/?h=orc-0.5>