Back-end
missing pieces

Simone Campanoni
simonec@eecs.northwestern.edu
Instruction selection is part of the backend
Register allocation after instruction selection

Total cost: 5

Optimum!

A register allocation

v1 *= 4
v2 <- v1
v2 += 5
v3 <- mem v1 0
Register allocation after instruction selection

lea $(5+v1*4)$, %v2
subq %v2, %v1
movq $0(\%v1)$, %v3

lea $(5+\%rax*4)$, %rbx
subq %rbx, %rax
movq $0(\%rax)$, %r10
movq %r10, $0(\%rsp)$

A register allocation:
- $v1 \rightarrow \%rax$
- $v2 \rightarrow \%rbx$
- $v3 \rightarrow \text{stack O}$
Register allocation after instruction selection

A register allocation

```
v1 -> rax
v2 -> rbx
v3 -> stack O
v4 -> r8
```

Peephole matching

```
lea (5+%rax*4), %rbx
subq %rbx, %rax
movq 0(%rax), %r10
movq %r10, 0(%rsp)
movq O(%rsp), %r8
```
Instruction selection is part of the backend

- Tracing and data layout
- Instruction selection
- Register allocation
- Code generation
- Peephole matching

IR → Back-end → Assembly
Peephole matching

• Basic idea: compiler can discover local improvements locally
  • Look at a small set of adjacent operations
  • Move a “peephole” over code & search for improvement

• Example: store followed by load

\[
\begin{align*}
\text{movq} & \ %r10, O(\%rsp) \\
\text{movq} & \ O(\%rsp), \ %r8
\end{align*}
\]

\[
\begin{align*}
\text{movq} & \ %r10, O(\%rsp) \\
\text{movq} & \ %r10, \ %r8
\end{align*}
\]
Are we happy now with the generated assembly?

Of course NOT!
The problem left

```assembly
lea (5+%rax*4), %rbx
subq %rbx, %rax
movq 0(%rax), %r10
movq %r10, 0(%rsp)
movq %r10, %r8
subq %r9, %r10
movq %r10, 0(%r11)
```

Better schedule of instructions

```assembly
lea (5+%rax*4), %rbx
subq %r9, %r10
subq %rbx, %rax
movq %r10, 0(%r11)
movq 0(%rax), %r10
movq %r10, 0(%rsp)
movq %r10, %r8
```
Instruction selection is part of the backend

Back-end

- Tracing and data layout
- Instruction selection
- Register allocation
- Code generation
  - Peephole matching
  - Instruction scheduling
- Assembly