Back-end
missing pieces

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

\[
\begin{align*}
\text{v1} &= \times 4 \\
\text{v2} &= -\text{v1} \\
\text{v2} &= +5 \\
\text{v3} &= \text{mem v1 0} \\
\text{v1} &= \times v1 \times 4 \\
\text{v2} &= +5 \\
\text{v3} &= \text{lea (5+\%v1*4), \%v2} \\
\text{v1} &= \text{subq \%v2, \%v1} \\
\text{v3} &= \text{movq 0(\%v1), \%v3}
\end{align*}
\]

Total cost: 5

Optimum!
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, O(%rsp)

A register allocation
v1 -> rax
v2 -> rbx
v3 -> stack O

Temporary register
Register allocation after instruction selection

lea (5+\%v1*4), \%v2
subq \%v2, \%v1
movq 0(\%v1), \%v3
movq \%v3, \%v4

A register allocation
v1 \rightarrow \text{rax}
v2 \rightarrow \text{rbx}
v3 \rightarrow \text{stack O}
v4 \rightarrow \text{r8}

lea (5+\%rax*4), \%rbx
subq \%rbx, \%rax
movq 0(\%rax), \%r10
movq \%r10, O(\%rsp)
movq O(\%rsp), \%r8

Peephole matching

Wait, I thought we found the optimum ...
Instruction selection is part of the backend

IR

Tracing and data layout → Instruction selection → Register allocation → Code generation

Back-end

Code generation → Peephole matching → 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

```plaintext
movq %r10, O(%rsp)
movq O(%rsp), %r8
```

Peephole matching

```plaintext
movq %r10, O(%rsp)  
movq %r10, %r8
```
Are we happy now with the generated assembly?

Of course NOT!
The problem left

Instruction scheduling

Better schedule of instructions