Register Allocation

mov a, [b+8] ⟶ mov edx, [ebp-k_b]
mov ecx, [edx+8]
mov [ebp-k_a], ecx

mov ecx, [edx+8]

live vars

```
mov b, a           ; edx  eax
mov c, [b * 4]     ; edx  edx
lea b, [c+1]       ; edx  edx
mov d, a           ; eax  eax
add d, b           ; eax  edx
mov eax, d         ; eax
ret
```

Live vars: a / a, b / a, c / a, b / b, d / d / eax / ∅

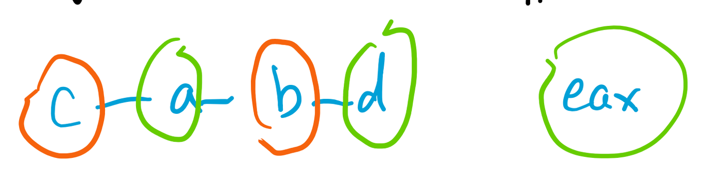
a ↦ eax
b ↦ edx
c ↦ edx
d ↦ eax

imul t ; multiplies eax and t, stores edx : eax

Should not allocate edx, eax to any var live-out from the instr.

Graph coloring.

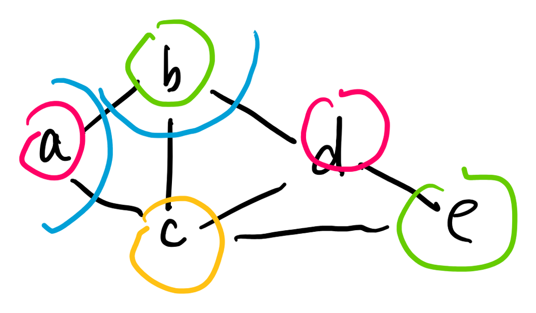
interference graphs. nodes = variables and registers
edges = nodes cannot be mapped to same register



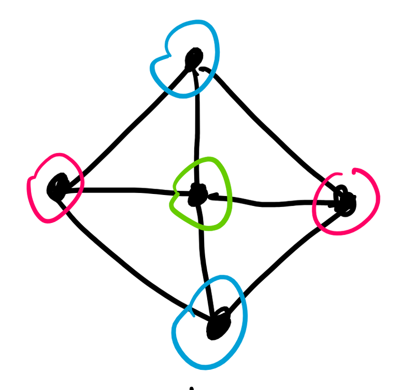
Kempe's heuristic (1880) / optimistic coloring

k colors available
low-degree nodes : degree < k
high-degree nodes : --- ≥ k

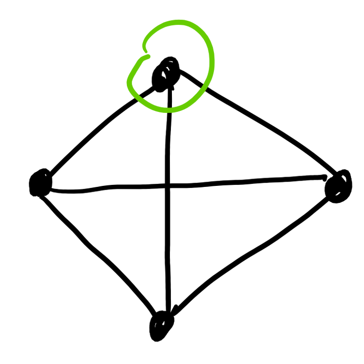
Alg. If ≤ k nodes left, done
If ∃ low-degree node :
color the rest of graph recursively
pick a color different from the neighbors.
Else
pick a high-degree node
color the rest of graph recursively
try to find color that works



k=3



3-colorable



not 3-colorable

How to spill ?

spill with fresh variables and retry reg. allocation.

Ex. add t_2, t_1 ; spill t_2 to [ebp-8]

mov t_{42}, [ebp-8]
add t_{42}, t_1
mov [ebp-8], t_{42}

t_{42} has short liveness range
⇓
generate little interference
⇓
easy to assign reg to t_{42}

x86.

	use	def
imul t	eax	eax, edx
jecxz	ecx	
jcxz	cx	
loop t	ecx	ecx
ret	eax	
call		all caller-saved regs.

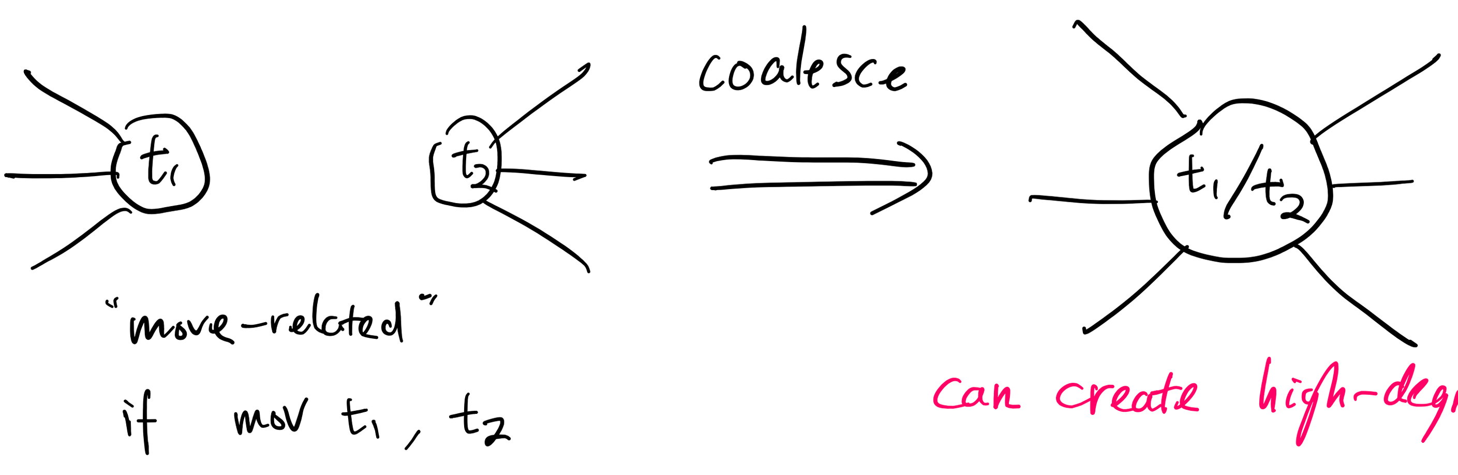
add pre-colored node to ..
add edges

cannot cut during coloring.

vars live-out from call interfere with caller-saved registers.
⇓
use callee-saved regs. live-out from call.
⇓
prefer caller-saved regs. for vars not live-out from call.

How to coalesce movs ?

Goal. reduce # of mov
Idea. delete "mov t_1, t_2" if t_1 ↦ r and t_2 ↦ r

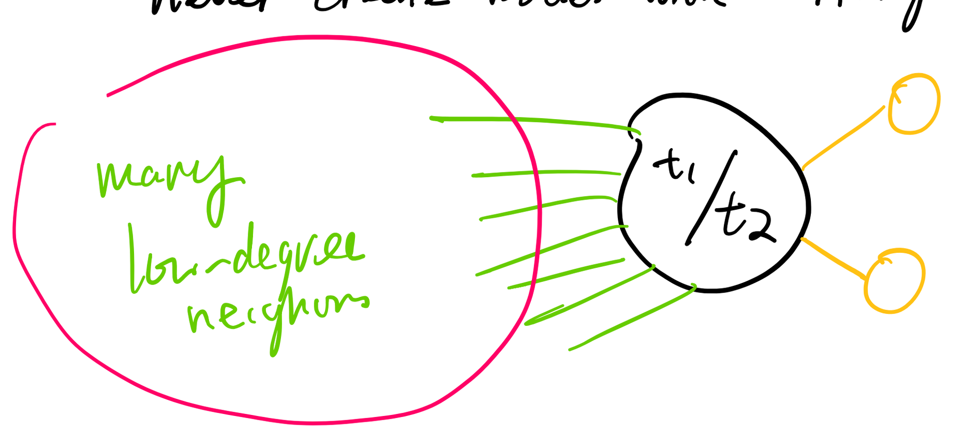


"move-related"
if mov t_1, t_2

can create high-degree node

Conservative coalescing.

never create nodes with ≥ k high-degree neighbors.



(k−1) high-degree neighbors

Chaitin's algorithm 1980s

0. LVA and build interference graph
1. Chop off low-degree, non-move-related nodes & push onto stack
   all nodes are high-degree or move-related
2. Conservatively coalesce mov-related nodes
   may create new low-degree, non-mov-related nodes ⇒ rerun from 1
   all nodes are high-degree or mov-related
   move-related nodes cannot be conserv. coalesced.
3. Mark low-degree, mov-related nodes as "non-move-related"
   ⇒ rerun from 1
   all nodes are high-degree
4. Choose a high-degree node for potential spill, remove & push
   ⇒ rerun from 1
   no nodes left in graph
5. pop nodes from stack & try assigning colors.
   - If node is pushed in step 1 (Kempe's), find a color
   - If node is pushed in step 4 (optimistic), try finding a color
     If failure, rewrite code for spilling ⇒ rerun from 0.