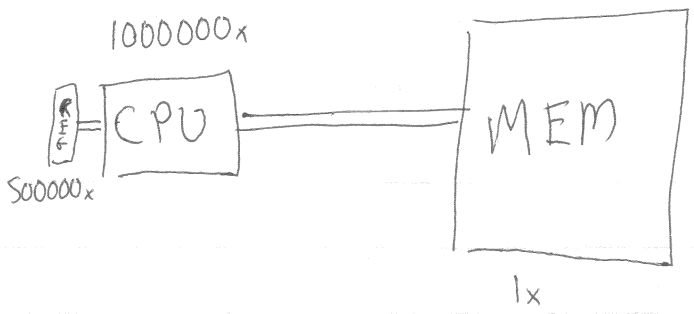
4-1



$$\frac{M(P) = M(P') \quad V(P) < V(P')}{OPT(P \Rightarrow P')}$$

M = meaning ≠ behavior

V = value

~~FCX=1~~

v=1, w=46, x=v+7, y=4+x, z=x+w, ret z+(-y);

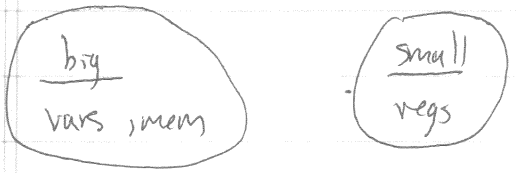
Change mem-ref into reg-ref

x*

-inf = 0	(program (v w x y z t1 t2)
1	(movq 1 v) ∅
2	(movq 46 w) v
3	(movq v x) w, v
4	(addq 7 x) w, x
5	(movq x y) w, x
6	(addq 4 y) y, w, x
7	(movq x z) y, w, x
8	(addq w z) z, y, w
9	(movq y t1) z, y
10	(negq t1) t1, z
11	(movq z t2) t1, z
12	(addq t1 t2) t1, t2
13	(movq t2 rax)) t2
+inf = 14	∅

assign mapped all vars to mem

regalloc map some vars to regs
and some to mem



1x naive: all mem

100x less naive: first |Registers| get reg, else mem

10000x better: some ~~variable~~ pairs of variables don't overlap in usefulness, so, they can share registers

30000x best: range-splitting (move a lightly used variable to mem when lightly used)

Liveness = when a variable is needed
: a range of program points

Interference = when two variables are needed at once
: an overlapping of ranges (i.e. being in the same liveness-set for one instruction)

Spilled = a variable not in a register

L_i = the liveness set of instruction i

La_i = liveness ~~before~~ after i

Lb_i = liveness ~~after~~ before i

writes-to
W : instr → vars

R : instr → vars
reads-from

$L_{b0} = \emptyset \quad La_0 = \emptyset \quad La_{i,n} = \emptyset$

$La_{i,k} = Lb_{i,k+1}$

$Lb_{i,k} = (La_{i,k} - W(k)) \cup R(k)$

$W(\text{addq } s\ d) = \{d\} \quad W(\text{movq } s\ d) = \{d\}$

$R(\text{addq } s\ d) = \{s, d\} \quad W(\text{~~addq~~ } s\ d) = \{s\}$

Interferences = live at same time

u interferes with v iff

$\exists i.$ $\{u, v\} \subseteq L(i)$

1. for u = 0 to N variables
2. for v = u+1 to N variables
3. for i = 0 to eop(13)
   if $u \in L(i) \wedge v \in L(i)$
   m[u][v] = 1
   break ;

$n \cdot \frac{n}{2} \cdot k \leq k \cdot \frac{k}{2} \cdot k = O(k^3)$

Interference graph $I = (Vars, E)$ $E\ u\ v$ iff u interfers with v

for i = 0 to EOP(13)

match instr i with

(movq s d) => $\forall v \in L(i)$, add (d,v) to E
pushq s
popq d
unless v = d or v = s

(addq s d) => $\forall v \in L(i)$, add (d,v) to E
subq
unless v = d

negq => like add but w/o s