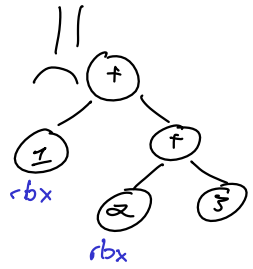
Register Allocation

very approximate

* Access to a register : ~1 clock cycle
  - to L1 cache : ~4 clock cycles
  - to L2 cache : ~10 clock cycles
  - to L3 cache : ~40 clock cycles
* Access to RAM : "~100-300 cycles"



$$\left(\{ x \mapsto [rbp-8] \}, -16 \right)$$

rax
r10
r11
rdi
r8
r9
rsi
r12
r13
r14
r15

$$\left(\{ x \mapsto [rbp-8] \}, [[rbp-16], [rbp-24], \ldots] \right)$$

$$\left(\{\}, [rbx, r8, r9, r12, \ldots, r15, [rbp-8], \ldots] \right)$$

(1) Greedy algorithms

1+2

```
mov rax, 2
mov [rbp-8], rax
mov rax, 4
mov [rbp-16], rax
mov rax, [rbp-8]
add rax, [rbp-16]
```

```
mov rax, 2
mov rbx , rax
mov rax, 4
mov r8 , rax
mov rax, rbx
add rax, r8
```

Intermediate Representation (IR) (ex. LLVM)

```
mov rax, 2
mov place1 , rax
mov rax, 4
mov place2, rax
mov rax, place1
add rax, place2
```

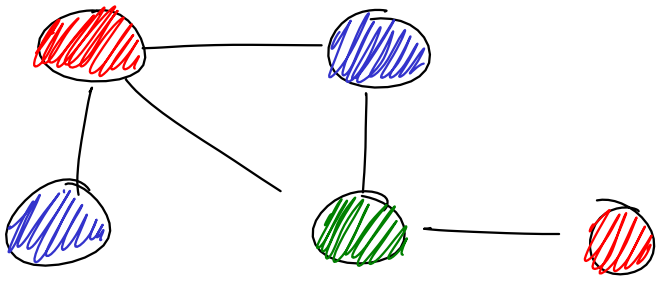
(2) Graph Coloring

Given a graph $G = \langle V, E \rangle$, choose a mapping $M : V \to Color$ s.t. if $\langle V_1, V_2 \rangle \in E$, then $M(V_1) \neq M(V_2)$.

Problem:

pick M to use smallest # of colors

NP-complete !!

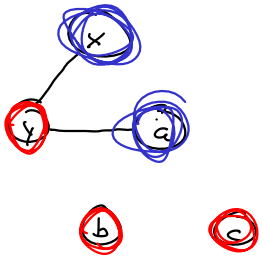


Vertices are variables

Edges are "interference": coexistence

Colors are storage locations.

```
let x = 1 in
let y = x+2 in
let a = x*3 in
let b = y+a in
let c = b+1 in
c+2
```

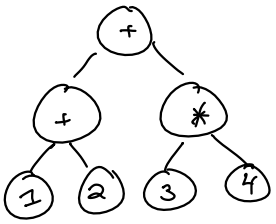


doesn't cover temp vars!

(3) Linear Scanning (similar to (1))

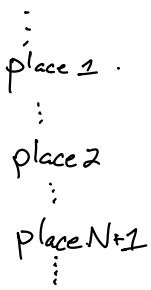
~ Greedy + swapping.

Suppose 2 storage: p1 & p2




```
                mov  rax, 2
                mov  p1, rax
                mov  rax, 4
                mov  p2, rax
p2=1+2          add  p2, p1
                mov  rax 6
                mov  p1, rax
                mov  [rbp-8], p2
[rbp-8]=1+2     mov  rax, 8
                imul  rax, p1
                mov  p1, rax
                mov  rax, p1
                mov  p2, [rbp-8]
                add  rax, p2
```

N storage



1. Not as fast (at runtime) as graph coloring (~~120%~~ slower)
2. Much faster (at compile time)