# Register Allocation

eax – used for returns, etc.

ebx – used for validation

$\underbrace{\text{ecx, edx, esi, edi}}_{\text{fast storage}}$

## Strategies

- Allocate registers first
- Use registers as cache
- Use registers for freq. accessed info

## Environment

{ x ↦ [ebp -4] } , [ecx; edx; esi; edi] @ [ [ebp-8]; [ebp-12]; ... ]

## Greedy Allocation

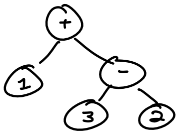
1+(3-2)

```
mov eax, 2
mov ecx, eax
mov eax, 6
mov edx, eax
mov eax, 4
mov esi, eax
mov eax, edx
sub eax, esi
mov edx, eax
mov eax, ecx
add eax, edx
```

temp alloc 3 in edx

## Linear scanning

Example : limit to ecx, edx



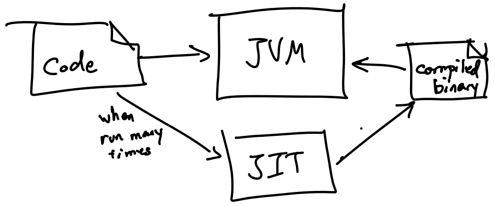
When all registers are allocated, "spill" into memory

```
mov eax, 2
mov ecx, eax
mov eax, 6
mov edx, eax
mov eax, 4
mov [esp-4], ecx   ; evict 1st temp
mov ecx, eax
mov eax, edx
sub eax, ecx
mov edx, eax
mov ecx, [esp-4]   ; restore 1st temp
mov eax, ecx
add eax, edx
```

{ temp1 ↦ ecx
temp2 ↦ edx }

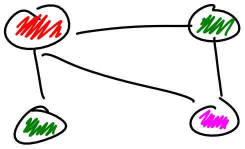
{ temp1 ↦ [ebp-4]
temp2 ↦ edx
temp3 ↦ ecx }

## Just-In-Time

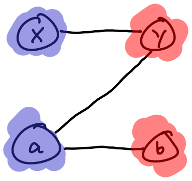


# Graph Coloring Allocation

Vertices : storage
Edges : interference



```
let x = 5 in              ∅
let y = 7 in             {x}
let a = x+1 in           {y}
let b = y+2 in           {a}
a-b                     {a,b}
```



Graph coloring is NP-complete

Colors are registers (or other storage)

Linear scanning : good for JIT ; about 12% slower than graph coloring
Graph coloring : slower but produces faster code