R1

20 January 2022 03:10

exp ::= int | (read) | (- exp) | (+ exp exp) | var | (let ([var exp]) exp)

R_1 ::= exp

concrete Syn

assign body

(let ([x (+ 10 20)]) (+ x 30)) → 60

x = (+ 10 20)

(+ x 30) => (+ 30 30)

(let [var RHS] body)

(let [x (+ 10 20)] x)

(+ x 10)

(let [x:1 (+ 10 20)] (+ (let [x:2 (+ 2 x:1)] x:2) x:1))

x = 10+20

↳ x = 2+3

↳x

↳ ~~x+10~~ ~~x+x~~ 5 + x

↳ 5 + 30

↳ 35

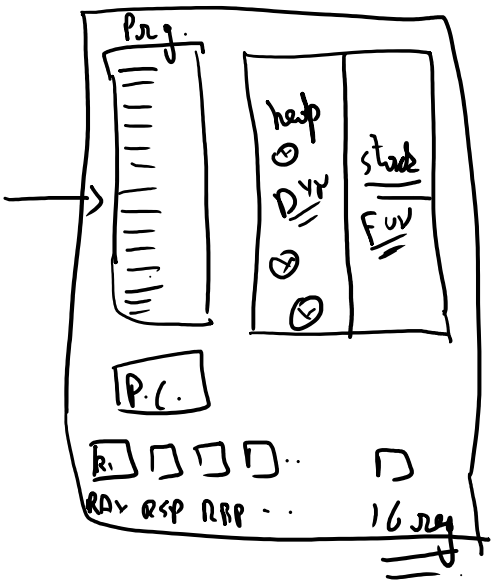
A.S. R0

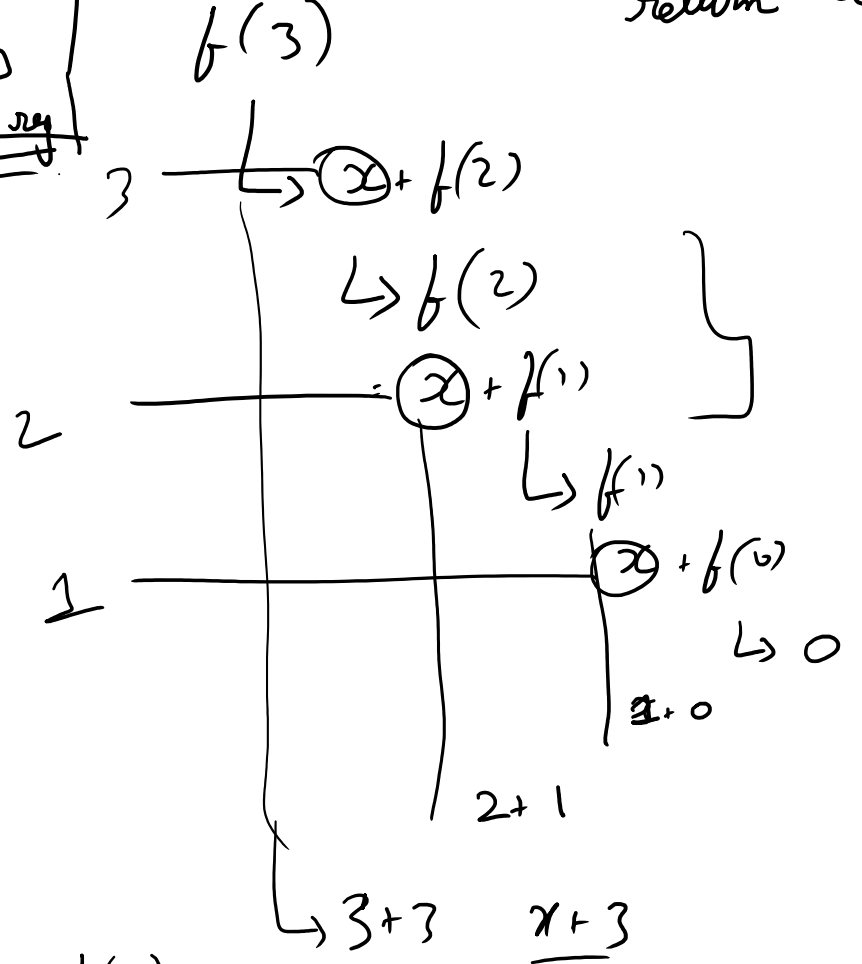
exp ::= (Int int) | (Prim 'read ()) | (Prim '- exp) | (Prim '+ (exp exp)) | ~~(Var var)~~

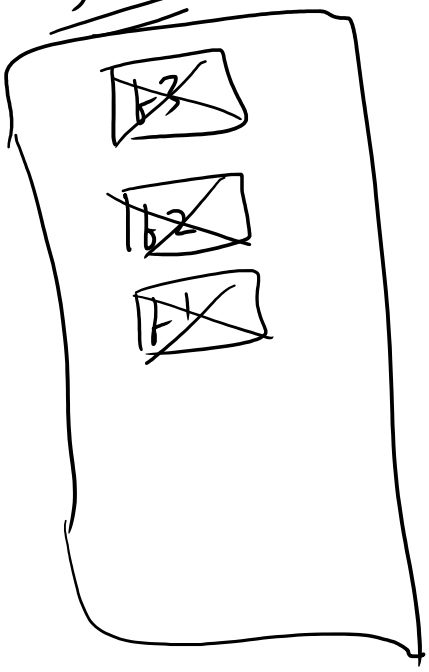
(Var var) | (Let var exp exp)

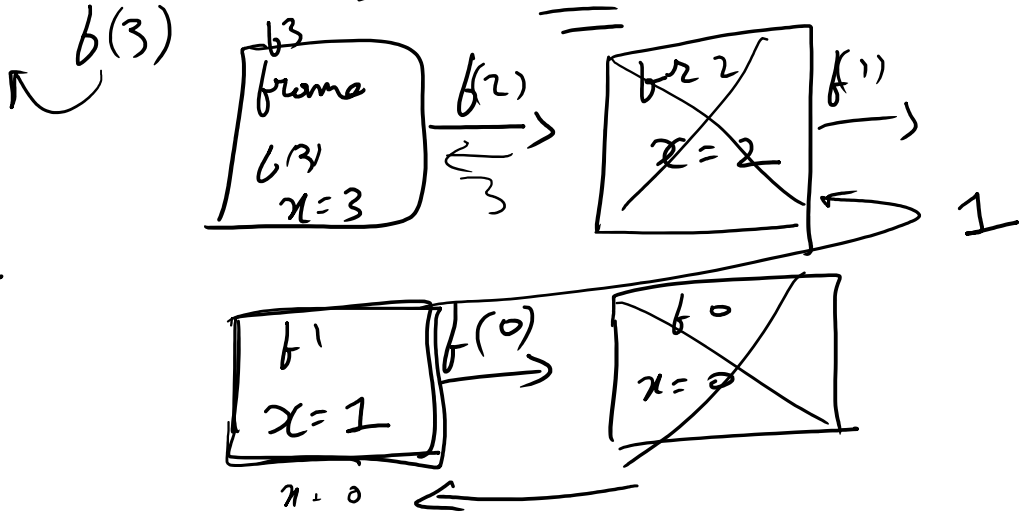
10 BYTES

```
int f(int n){
if (n==0){
return 0;
else
return n+f(n-1);
```









C, PY, RAC.

P implicit

* impure

x86 → partially implement }

X86 instr .

20 January 2022 03:47

reg ::= rsp | rax | rbp ... | r15

arg ::= $int | %reg | int(%reg)

instr ::= addq arg, arg (src, dest)

negq arg

movq arg, arg

callq label

pushq arg

popq arg

retq

prog ::= .global main

$8, $32 (32)

%rax %rbi

-8(%rbp)

dest = src + dest

dest = -dest

movq %rax %rbi

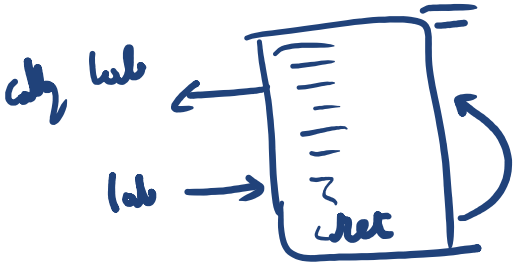
%rax -8(%rbp)

→ func.

pushq %rax

popq %rbp

main: instr*



X86 example

20 January 2022 03:56

(+10 45)

```
.global main
main:
        movq $10 %rax
        movq $45 %rbi
        addq $45 %rax
        movq %rax %rbi
        call print_int
        movq $0 %rax
        retq
```

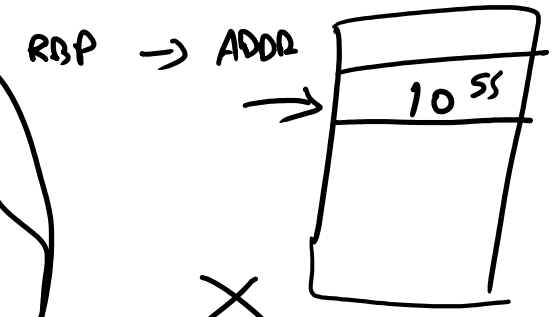
RAX is return

```
f1:
        movq $10 %rax
        addq $45 %rax
        retq
```

RAX

RBP → ADDR - 8

```
movq $10  -8(%rbp)
addq $45  -8(%rbp)
movq -8(%rbp) %rax
retq
```



Regs → Reg. allocate

R1 v x86

20 January 2022 04:09

R1 —compiler→ x86

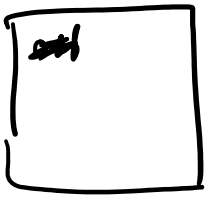
→(+ 10 (+ 1 2))
(+ 2 3)
add

complex nested

remove-complex-op

atomic exp

(+ 10 (+ 1 2)) —process→



10 x86

{ shadowing variables

make unique

regs & mem.

uniquify

removed

error handle

difficult x86

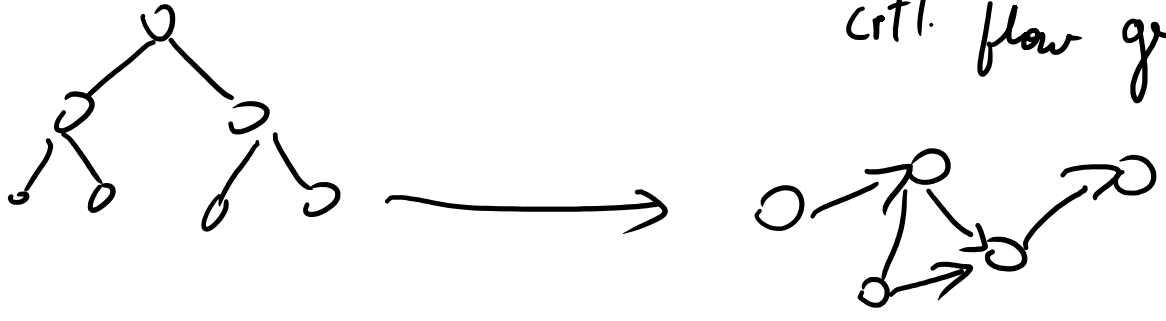
arb. variable —'assign home'→ 16 regs. arb memory

depth first exe tree

explicate control

seq jump

ctrl. flow graph



(+ 10 42)

addq %rax %rbp

(+ 10 42)
↳ 52

2 i/ps
1 o/p

select inst →

compiler

addq %rax %rbp

addq src. dst.
dst updated

passes

20 January 2022 04:30