16-1/ f=q · ∀x. fx = g x Ry -> R5 e:= ... | (lambda var ([var:4]...):4 e) / 7 7 7 it for any ret (() fac ([n: 564]) = 564 (if (= 0 n) 1 (* n (fac (-n ())))) for (m+ i=0; ic 40; i++) & C for (int j=0; j < 60; j++) { CIR ...

U (let/ec return 16-2/ (+ (if (...) (return 0)) (letlec ret, & (+ (letlec retz & ened, 1, M[r+>f4][a+4]...+e:ret (retz-A, 17 + (1 r ([a:4]...) n+ e): (+y ... → n+) (let x:= xe in b) => 50% a let 3 (() (x) b) xe)Op+ (app le aes) => (let ([a ae] ...) body) if fe ngs "known" tenous mens lakel or (1 (....)

16-3/ New pass: convent-to-closure cloconv: R5 > Ry nas no d $(\lambda (xy) (+xy))$ => (vector (fun-ref nf17)) on side: nf 17 = define (x y) $(+ \times y)$ (f 12) => (app (vector-ref f 0) ((let = = 4 in $(((5 \times 4) (\chi) (\chi)))$ (define (nf 29 [v: 7[x:564]):564 (let Z := (vector-ref v 1) in ((let z:= 4 in (vector nf29 Z)) in (app (rector-ref (0) (5))

16-4) type of closure M binding rec M existing top-levels M rec ; () r (a ...) e) => (define (nf [r][a]...) e) the name used fur 'rec' (define (sub 1 [x: S64]): S64 (-x 1)) (Sub 1 4) => (app (vector-ref sub 1 0) (for-ref f) (veltor (for-ref f)) (Lefre Geren?) ... odd?)
(before GdJ?) ... even?) (104) () ~ ([x:564]):564 (+ x Z)) =7 (vector nf 42 z) : Fallete V nf42: F F \ V x S(y \Rightarrow S(y) = (Vector Trubble St) V = Vector F 564 X 564 > 564

16-5/ cloconu: e 7 e x (13+ of defs)
<u></u>
are all the
cloons (+ el er) lambda bodies
= (+ ei ee')
x (1bL ++ 1bR)
Where (el', 1bL) = cloconv el
(eR', lbR) = cbconv eR
more instructions - more depon arch
in Struction scheduling turns "A" rode
like Z= a * 12 2
=7 shiga, Z
mong q, z
opdimited 1:01 => ! = (0, 50) :++
=7 ;= [0,80] ++=8
$(+(-\times 1) 2) \implies X$
(+ x (-1))
(+ (-1) x)
(+ (+1-1) x)) (+ 0x)