| 8-11 | Inlining memores ton calls |
|--|--|
| | ? stack space (for non-tail coalls) |
| | + removes jumps (ret is an indirect, to direct) |
| | + no caller/ee van saving |
| | - Fin grows, big for => lots of registers=> register pressure |
| SEPTION CONTINUES AND CONTINUE | - cole size hurts i-cache |
| voltakalinis salamakajajahalaikatinensas teritikinkaini, juura yawa, gayararendameni ya ciri ciri ciri ciri ciri ciri ciri cir | (matters most in loop bodies) |
| esseg (ES) to our organisate (Es) in Wellium and Arman and Equipment (Es) and Arman and Arman and Equipment (Es) and Es) and E | t exposes other optimizations |
| | (define (fx) (if x 42 ga)) |
| egy megeneten myske gypen rigest det mention bliner dinn unerzon. Javen husbi din 100 till 100 400 600 400 400 | (+ (+ (+ the) (+ fal)) (+y)) |
| | - code size makes compile longer |
| tersindiges jurigidige filt de glight de de glight de grand de grand de grand de grand de grand de grand de glight de glight de grand de g | - (define (fac n) |
| | Cif (= n 0) 1 (* n (fac (S.vb) n)))) |
| e en gladely (tryle (K) | (fac 5) |
| a 22 a 25 | |
| | Program is a set |
| | Optimize; : Prog > Prog |
| | Po is user-input |
| O _* = | compiler, (Po) = Optimizes (Oz (O; (Oo Po)))) |
| | Compiler F (P) = Px |
| | |
| | $O_{*}(P_{X}) = P_{X}$ alwayse |
| | |
| | p optimize p run a Jeompile compile) p run a |
| | Jeompile compile) |
| 6 | X, 7 X Subject of the |
| | run |
| | run: prog > ans |
| | opt(p) = p run': prog => uns, cost |
| | |
| | |

| 8-7/ granate linear time |
|--|
| polyvariant - every funcall is rinlined independenty |
| monument - Forms are Mined, not for calls |
| |
| online - fast enough to be interlead al rest |
| of compile |
| context-sensitive - may the fun result is used |
| changes the optimization |
| $(\leq (fx) \neq)$ |
| (vector-ref (f x) 3) |
| boolean, effect, value |
| demand-driven - function bodies are not optimized |
| until called |
| " flor insensitie" |
| (define V (vector add 1)) [let [old (ur V 0)]) |
| (14) COI W |
| (vector-set/VoCif X add subi))) (XX, (+1 (old X)))) |
| (flip! +) (flip! f) (flip +) |
| ((rector-ref V O) Z) |
| e: = (Lonst c) (ref x) (primref p) |
| (if e e e) (seg e e) (assign x e) |
| (/ (x) e) ((etrec ([x e]) e) |
| (call e e) x + Var |
| (|
| I: Expression e -> (ontext >> Env p -> (ontinuation K -> Store or > e |
| p = Env = Var = Var Var Var := (Id x O perand 1, Var Flags, Locx) |
| |
| K = (ont = Exp => Store > Exp |
| 5 = Store = (Locx -> Van Flags) x (Locy -> Context Flags) x (Loce -> Exp_) |
| Var Flags & Eref, assign 3. Content Flags = Einlined3 |
| Y + Context = Test Effect Value App (Operand, Y, Locy) |
| Operand = Opnd (Exp, Env, Loce) |
| |