

module test where

open import lib  
open import source  
open import target  
open import compiler

-- term-0 : do nothing  
term-0 : · ⊢ comm  
term-0 = Skip -- source term  
result-0 = compile-closed term-0

test-0 : result-0 ≡ stop -- target term  
test-0 = refl

-- term-1 : x := 2  
term-1 : · ⊢ comm  
term-1 =  
 NewVar  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Lit (pos 2)))  
result-1 = compile-closed term-1  
  
test-1 : result-1 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-dec  
 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 2)))  
 (adjustdisp-dec 1 (s≤s z≤n) stop))  
test-1 = refl

-- term-2 : x := (λ a. a) -4  
term-2 : · ⊢ comm  
term-2 =  
 NewVar  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (App  
 (Lambda (Var Zero))  
 (Lit (negsuc 3))))  
result-2 = compile-closed term-2  
  
test-2 : result-2 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (negsuc 3)))  
 (adjustdisp-dec 1 (s≤s z≤n) stop))  
test-2 = refl

-- term-3 : x := (λ a. (λ b. a + b) 2) 3  
term-3 : · ⊢ comm  
term-3 =  
 NewVar  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (App  
 (Lambda  
 (App  
 (Lambda  
 (Plus  
 (Var (Suc Zero))  
 (Var Zero)))  
 (Lit (pos 2))))  
 (Lit (pos 3))))  
result-3 = compile-closed term-3  
  
test-3 : result-3 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-binary  
 (s-lit (pos 3))  
 BPlus  
 (s-lit (pos 2)))  
 (adjustdisp-dec 1 (s≤s z≤n) stop))  
test-3 = refl

-- term-3' : x := 3 + 2  
term-3' : · ⊢ comm  
term-3' =  
 NewVar  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Plus  
 (Lit (pos 3))  
 (Lit (pos 2))))  
result-3' = compile-closed term-3'  
  
test-3' : result-3' ≡ result-3  
test-3' = refl

-- term-4 : x := -3; y := (λ a. x) 2  
term-4 : · ⊢ comm  
term-4 =  
 NewVar  
 (Seq  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Lit (negsuc 2)))  
 (NewVar  
 (Assign  
 (Sub (Var (Suc Zero)) var-≤:-acc)  
 (App  
 (Lambda (Sub (Var (Suc (Suc Zero))) var-≤:-exp))  
 (Lit (pos 2))))))  
result-4 = compile-closed term-4  
  
test-4 : result-4 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (negsuc 2)))  
 (assign-inc 1  
 (l-var ⟨ 0 , 1 ⟩ (≤-d (s≤s z≤n)))  
 (r-s (s-lit (pos 0)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-l (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n)))))  
 (adjustdisp-dec 1 (s≤s z≤n)  
 (adjustdisp-dec 1 (s≤s z≤n) stop))))))  
test-4 = refl

-- term-5 : x := 2; x := -x  
term-5 : · ⊢ comm  
term-5 =  
 NewVar  
 (Seq  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Lit (pos 2)))  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Neg (Sub (Var Zero) var-≤:-exp))))  
result-5 = compile-closed term-5  
  
test-5 : result-5 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 2)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-unary UNeg (s-l (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n)))))  
 (adjustdisp-dec 1 (s≤s z≤n) stop)))  
test-5 = refl

-- term-5' : x := 2; skip; x := -x  
term-5' : · ⊢ comm  
term-5' =  
 NewVar  
 (Seq  
 (Seq  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Lit (pos 2)))  
 Skip)  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Neg (Sub (Var Zero) var-≤:-exp))))  
result-5' = compile-closed term-5'  
  
test-5' : result-5' ≡ result-5  
test-5' = refl

-- term-6 : x := 2; x := -x + 1  
term-6 : · ⊢ comm  
term-6 =  
 NewVar  
 (Seq  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (Lit (pos 2)))  
 (Assign  
 ((Sub (Var Zero) var-≤:-acc))  
 (Plus  
 (Neg (Sub (Var Zero) var-≤:-exp))  
 (Lit (pos 1)))))  
result-6 = compile-closed term-6  
  
test-6 : result-6 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-dec 0 z≤n  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 2)))  
 (assign-inc 1  
 (l-var ⟨ 0 , 1 ⟩ (≤-d (s≤s z≤n)))  
 (r-unary UNeg (s-l (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n)))))  
 (assign-dec 1 (s≤s z≤n)  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-binary  
 (s-l (l-var ⟨ 0 , 1 ⟩ (≤-d (s≤s z≤n)))))  
 BPlus  
 (s-lit (pos 1)))  
 (adjustdisp-dec 1 (s≤s z≤n) stop)))  
test-6 = refl

-- term-7 : x := (λ a. -a + 1) 2  
term-7 : · ⊢ comm  
term-7 =  
 NewVar  
 (Assign  
 (Sub (Var Zero) var-≤:-acc)  
 (App  
 (Lambda  
 (Plus  
 (Neg (Var Zero))  
 (Lit (pos 1))))  
 (Lit (pos 2))))  
result-7 = compile-closed term-7  
  
test-7 : result-7 ≡  
 assign-inc 1  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-s (s-lit (pos 0)))  
 (assign-inc 1  
 (l-var ⟨ 0 , 1 ⟩ (≤-d (s≤s z≤n)))  
 (r-unary UNeg (s-lit (pos 2)))  
 (assign-dec 1 (s≤s z≤n)  
 (l-var ⟨ 0 , 0 ⟩ (≤-d z≤n))  
 (r-binary  
 (s-l (l-var ⟨ 0 , 1 ⟩ (≤-d (s≤s z≤n)))))  
 BPlus  
 (s-lit (pos 1)))  
 (adjustdisp-dec 1 (s≤s z≤n) stop)))  
test-7 = refl