```
(define (D exp)
  (cond ((number? exp) (%num exp))
        ((variable? exp) (%var exp))
        ((lambda? exp)
         (%lambda (lambda-variable exp)
            (D (lambda-body exp))))
        ((if? exp)
         (%if (D (if-condition exp))
              (D (if-consequent exp))
              (D (if-alternative exp))))
        ((+? exp)
         (%+ (D (op-arg1 exp))
             (D (op-arg2 exp))))
        ((*? exp)
         (%* (D (op-arg1 exp))
             (D (op-arg2 exp))))
        (else
         (%call (D (call-operator exp))
                (D (call-operand exp))))))
```

Monad	Action $T(A) =$
Identity	A
Lists	$List(A)$
Lifting	$1 \to A$
Environments	$Env \to A$
Stores	$Sto \to A \times Sto$
Exceptions	$A + X$
Monoids	$A \times M$
Continuations	$(A \to Ans) \to Ans$
Resumptions	$fix(X)\ (A + X)$