theory *SALSemantics* = *Semantics* + *TermCodegen*:

1 SAL Semantics

datatype
  $tval = ILLEGAL \mid NAT\ nat$

types
  $val\ \ = nat$ — or anything else, *nat* used in examples
  $loc\ \ = nat$
  $state = loc \Rightarrow tval$ — value and type information are stored together

constdefs
```
  lift :: (val ⇒ val ⇒ val) ⇒ tval ⇒ tval ⇒ tval
 lift f a b ≡ (case a of
    ILLEGAL ⇒ ILLEGAL
  | NAT m ⇒ (case b of
      ILLEGAL ⇒ ILLEGAL
    | NAT n ⇒ NAT (f m n)))
```

```
datatype instr = SET loc val |
                 ADD loc loc |
                 INC loc |
                 JMPEQ loc loc nat |
                 JMPB nat
```

types $SALstate = nat \times state$
types $SALform = SALstate \Rightarrow bool$
types $SALprogram = (instr \times (SALform\ option))\ list$

constdefs $cmd{::}SALprogram \Rightarrow (nat \Rightarrow instr\ option)$
$cmd\ p \equiv \lambda\ i.\ if\ (i < length\ p)\ then\ Some\ (fst\ (p!i))\ else\ None$

constdefs $domC{::}SALprogram \Rightarrow (nat\ list)$
$domC\ p \equiv upt\ 0\ (length\ p)$

constdefs $an{::}\ SALprogram \Rightarrow (nat \Rightarrow SALform\ option)$
$an\ p \equiv \lambda\ i.\ if\ (i < length\ p)\ then\ snd\ (p!i)\ else\ None$

consts
  $step{::}SALprogram \Rightarrow SALstate \Rightarrow SALstate\ option$
primrec
```
 step p (i, m) = (case (cmd p i) of
    None ⇒ None
  | Some ins ⇒ (case ins of
      SET x n ⇒ Some (i + 1, m[x ↦ NAT n])
    | ADD x y ⇒ Some (i + 1, m[x ↦ lift (op +) (m x) (m y)])
    | INC x   ⇒ Some (i + 1, m[x ↦ lift (op +) (m x) (NAT 1)])
    | JMPEQ x y t ⇒ (if m x = m y then Some (i + t, m) else Some (i + 1, m))
    | JMPB t ⇒ Some (i − t, m)))
```

constdefs *effS*:: *SALprogram* ⇒ ((*nat* × (*loc* ⇒ *tval*)) × (*nat* × (*loc* ⇒ *tval*)))
set
effS (*p*::*SALprogram*) ≡ {(*s*::*SALstate*,*s'*::*SALstate*). (*step p s* = *Some s'*) }
end