Lemma init_goodis

Lemma init_goodis:
forall S O I lasrt sd init,
init S O ->
side_condition I lasrt sd init init_lg ->
good_is_S S.
Proof.

```
1 subgoal, subgoal 1 (ID 2719)
 ============================
   forall (S : osstate) (O : osabst) (I : Inv) (lasrt : LocalInv)
     (sd : ossched) (init : osstate -> osabst -> Type),
   init S O -> side_condition I lasrt sd init init_lg -> good_is_S S
```

intros.

```
1 subgoal, subgoal 1 (ID 2727)
  S : osstate
  O : osabst
  I : Inv
  lasrt : LocalInv
  sd : ossched
  init : osstate -> osabst -> Type
  X : init S O
  H : side_condition I lasrt sd init init_lg
  ============================
   good_is_S S
```

unfolds in H.

```
1 subgoal, subgoal 1 (ID 2728)
 S : osstate
 O : osabst
 I : Inv
 lasrt : LocalInv
 sd : ossched
 init : osstate -> osabst -> Type
 X : init S O
 H : GoodI I sd lasrt /\
     (forall (S : osstate) (O : osabst),
      init S O -> initst S O I lasrt init_lg /\ eqdomSO S O)
 ============================
  good_is_S S
```

mytac.

```
1 subgoal, subgoal 1 (ID 2733)

 S : osstate
 O : osabst
 I : Inv
 lasrt : LocalInv
 sd : ossched
 init : osstate -> osabst -> Type
 X : init S O
 H : GoodI I sd lasrt
 H0 : forall (S : osstate) (O : osabst),
      init S O -> initst S O I lasrt init_lg /\ eqdomSO S O
 ============================
  good_is_S S
```

apply H0 in X.

```
1 subgoal, subgoal 1 (ID 2735)

  S : osstate
  O : osabst
  I : Inv
  lasrt : LocalInv
  sd : ossched
  init : osstate -> osabst -> Type
  X : initst S O I lasrt init_lg /\ eqdomSO S O
  H : GoodI I sd lasrt
  H0 : forall (S : osstate) (O : osabst),
       init S O -> initst S O I lasrt init_lg /\ eqdomSO S O
  ============================
   good_is_S S
```

destruct X.

```
1 subgoal, subgoal 1 (ID 2742)

  S : osstate
  O : osabst
  I : Inv
  lasrt : LocalInv
  sd : ossched
  init : osstate -> osabst -> Type
  H1 : initst S O I lasrt init_lg
  H2 : eqdomSO S O
  H : GoodI I sd lasrt
  H0 : forall (S : osstate) (O : osabst),
       init S O -> initst S O I lasrt init_lg /\ eqdomSO S O
  ============================
   good_is_S S
```

clear H0 H2 H.

```
1 subgoal, subgoal 1 (ID 2743)

  S : osstate
  O : osabst
  I : Inv
  lasrt : LocalInv
  sd : ossched
  init : osstate -> osabst -> Type
  H1 : initst S O I lasrt init_lg
  ============================
   good_is_S S
```

induction H1.

```
  2 subgoals, subgoal 1 (ID 2782)

    sd : ossched
    init : osstate -> osabst -> Type
    S : env * cltenvs * mem * language.isr * ltaskstset
    O : osabst
    G : env
    envs : cltenvs
    M : mem
    isr : language.isr
    lst : ltaskstset
    E : env
    auxs : localst
    I : Inv
    pa : tid -> list logicvar -> asrt
    t : addrval
    lg : list logicvar
    H : S = (G, envs, M, isr, lst)
    H0 : envs = sig t E
    H1 : lst = sig t auxs
    H2 : get O curtid = Some (oscurt t)
    H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
    ============================
     good_is_S S

  subgoal 2 (ID 2809) is:
   good_is_S S
```

subst.

```
2 subgoals, subgoal 1 (ID 2821)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  ============================
   good_is_S (G, sig t E, M, isr, sig t auxs)

subgoal 2 (ID 2809) is:
 good_is_S S
```

unfolds.

```
2 subgoals, subgoal 1 (ID 2822)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  ============================
   forall (t0 : tid) (tst : env * env * mem * language.isr * localst),
   projS (G, sig t E, M, isr, sig t auxs) t0 = Some tst ->
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f

subgoal 2 (ID 2809) is:
 good_is_S S
```

intros.

```
  2 subgoals, subgoal 1 (ID 2825)

    sd : ossched
    init : osstate -> osabst -> Type
    O : osabst
    G : env
    M : mem
    isr : language.isr
    E : env
    auxs : localst
    I : Inv
    pa : tid -> list logicvar -> asrt
    t : addrval
    lg : list logicvar
    H2 : get O curtid = Some (oscurt t)
    H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
    t0 : tid
    tst : env * env * mem * language.isr * localst
    H : projS (G, sig t E, M, isr, sig t auxs) t0 = Some tst
    ============================
     let (p, l) := tst in
     let (p0, _) := p in
     let (p1, _) := p0 in
     let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f

  subgoal 2 (ID 2809) is:
   good_is_S S
```

destruct tst.

```
  2 subgoals, subgoal 1 (ID 2833)

    sd : ossched
    init : osstate -> osabst -> Type
    O : osabst
    G : env
    M : mem
    isr : language.isr
    E : env
    auxs : localst
    I : Inv
    pa : tid -> list logicvar -> asrt
    t : addrval
    lg : list logicvar
    H2 : get O curtid = Some (oscurt t)
    H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
    t0 : tid
    p : env * env * mem * language.isr
    l : localst
    H : projS (G, sig t E, M, isr, sig t auxs) t0 = Some (p, l)
    ============================
     let (p0, _) := p in
     let (p1, _) := p0 in
     let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f

  subgoal 2 (ID 2809) is:
   good_is_S S
```

destruct p.

```
2 subgoals, subgoal 1 (ID 2843)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  p : env * env * mem
  i : language.isr
  l : localst
  H : projS (G, sig t E, M, isr, sig t auxs) t0 = Some (p, i, l)
  ============================
   let (p0, _) := p in
   let (_, _) := p0 in let (p1, _) := l in let (_, f) := p1 in good_is f

subgoal 2 (ID 2809) is:
 good_is_S S
```

destruct p.

```
2 subgoals, subgoal 1 (ID 2853)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  p : env * env
  m : mem
  i : language.isr
  l : localst
  H : projS (G, sig t E, M, isr, sig t auxs) t0 = Some (p, m, i, l)
  ============================
   let (_, _) := p in let (p0, _) := l in let (_, f) := p0 in good_is f

subgoal 2 (ID 2809) is:
 good_is_S S
```

destruct p.

```
2 subgoals, subgoal 1 (ID 2863)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  l : localst
  H : projS (G, sig t E, M, isr, sig t auxs) t0 = Some (e, e0, m, i, l)
  ============================
   let (p, _) := l in let (_, f) := p in good_is f

subgoal 2 (ID 2809) is:
 good_is_S S
```

destruct l.

```
2 subgoals, subgoal 1 (ID 2871)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  p : ie * is
  c : cs
  H : projS (G, sig t E, M, isr, sig t auxs) t0 = Some (e, e0, m, i, (p, c))
  ============================
   let (_, f) := p in good_is f

subgoal 2 (ID 2809) is:
 good_is_S S
```

destruct p.

```
2 subgoals, subgoal 1 (ID 2881)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : projS (G, sig t E, M, isr, sig t auxs) t0 =
      Some (e, e0, m, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 2809) is:
 good_is_S S
```

unfolds in H.

```
2 subgoals, subgoal 1 (ID 2882)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : match projD (G, sig t E, M) t0 with
      | Some m =>
          match get (sig t auxs) t0 with
          | Some n => Some (m, isr, n)
          | None => None
          end
      | None => None
      end = Some (e, e0, m, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 2809) is:
 good_is_S S
```

unfold projD in H.

```
2 subgoals, subgoal 1 (ID 2883)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : match
        match get (sig t E) t0 with
        | Some e => Some (G, e, M)
        | None => None
        end
      with
      | Some m =>
          match get (sig t auxs) t0 with
          | Some n => Some (m, isr, n)
          | None => None
          end
      | None => None
      end = Some (e, e0, m, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 2809) is:
 good_is_S S
```

assert (t0 = t / t0 <> t) by tauto.

```
H0 : t0 = t \/ t0 <> t
```

destruct H0;subst.

```
3 subgoals, subgoal 1 (ID 3031)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : match
        match get (sig t E) t with
        | Some e => Some (G, e, M)
        | None => None
        end
      with
      | Some m =>
          match get (sig t auxs) t with
          | Some n => Some (m, isr, n)
          | None => None
          end
      | None => None
      end = Some (e, e0, m, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

rewrite map_get_sig in H.

```
3 subgoals, subgoal 1 (ID 3037)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : match get (sig t auxs) t with
      | Some n => Some (G, E, M, isr, n)
      | None => None
      end = Some (e, e0, m, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

rewrite map_get_sig in H.

```
3 subgoals, subgoal 1 (ID 3047)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : Some (G, E, M, isr, auxs) = Some (e, e0, m, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

inverts H.

```
3 subgoals, subgoal 1 (ID 3174)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  e, e0 : env
  m : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H3 : forall ab : absop,
       (e, e0, m, i, (i0, i1, c), O, ab) |= init_cur I pa t lg
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

unfold init_cur in H3.

```
3 subgoals, subgoal 1 (ID 3175)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  e, e0 : env
  m : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H3 : forall ab : absop,
       (e, e0, m, i, (i0, i1, c), O, ab)
       |= INV I **
          (EX tp : type, GV OSTCBCur @ Tptr tp |-r-> Vptr t) **
          init_rdy pa t lg
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

unfold init_rdy in H3.

```
3 subgoals, subgoal 1 (ID 3176)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  e, e0 : env
  m : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H3 : forall ab : absop,
       (e, e0, m, i, (i0, i1, c), O, ab)
       |= INV I **
          (EX tp : type, GV OSTCBCur @ Tptr tp |-r-> Vptr t) **
          pa t lg **
          [|GoodLInvAsrt pa|] **
          OS [empisr, true, nil, nil] ** A_dom_lenv nil
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

lets Hx:H3 (spec_done None).

```
3 subgoals, subgoal 1 (ID 3185)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  e, e0 : env
  m : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H3 : forall ab : absop,
       (e, e0, m, i, (i0, i1, c), O, ab)
       |= INV I **
          (EX tp : type, GV OSTCBCur @ Tptr tp |-r-> Vptr t) **
          pa t lg **
          [|GoodLInvAsrt pa|] **
          OS [empisr, true, nil, nil] ** A_dom_lenv nil
  Hx : (e, e0, m, i, (i0, i1, c), O, END None)
       |= INV I **
          (EX tp : type, GV OSTCBCur @ Tptr tp |-r-> Vptr t) **
          pa t lg **
          [|GoodLInvAsrt pa|] **
          OS [empisr, true, nil, nil] ** A_dom_lenv nil
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

simpl in Hx.

```
3 subgoals, subgoal 1 (ID 3680)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  e, e0 : env
  m : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H3 : forall ab : absop,
       (e, e0, m, i, (i0, i1, c), O, ab)
       |= INV I **
          (EX tp : type, GV OSTCBCur @ Tptr tp |-r-> Vptr t) **
          pa t lg **
          [|GoodLInvAsrt pa|] **
          OS [empisr, true, nil, nil] ** A_dom_lenv nil
  Hx : exists M1 M2 M o1 o2 o,
       M = m /\
       join M1 M2 M /\
       o = O /\
       join o1 o2 o /\
       (exists M3 M4 M0 o3 o4 o0,
        M0 = M1 /\
        join M3 M4 M0 /\
        o0 = o1 /\
        join o3 o4 o0 /\
        (e, e0, M3, i, (i0, i1, c), o3, END None) |= getinv (I (S INUM)) /\
        ((exists M5 M6 M7 o5 o6 o7,
          M7 = M4 /\
          join M5 M6 M7 /\
          o7 = o4 /\
          join o5 o6 o7 /\
          (i0 = true /\ M5 = empenv /\ emposabst o5) /\
          (exists M8 M9 M10 o8 o9 o10,
           M10 = M6 /\
           join M8 M9 M10 /\
```

```
  o10 = o6 /\
  join o8 o9 o10 /\
  (exists x,
   (x 0 = true /\ M8 = empenv /\ emposabst o8) /\
   i = x /\ M8 = empenv /\ emposabst o8 \/
   (exists M11 M12 M13 o11 o12 o13,
    M13 = M8 /\
    join M11 M12 M13 /\
    o13 = o8 /\
    join o11 o12 o13 /\
    ((... /\ ...) /\ i = x /\ ... /\ ...) /\
    (e, e0, M12, i, (i0, i1, c), o12, END None) |= getinv (I 0))) /\
  (exists M11 M12 M13 o11 o12 o13,
   M13 = M9 /\
   join M11 M12 M13 /\
   o13 = o9 /\
   join o11 o12 o13 /\
   (exists x,
    (... = true /\ ... /\ ...) /\
    i = x /\ M11 = empenv /\ emposabst o11 \/
    (exists M14 M15 M16 o14 o15 o16, M16 = M11 /\ ... /\ ...)) /\
   (exists x,
    (... = true /\ ... /\ ...) /\
    i = x /\ M12 = empenv /\ emposabst o12 \/
    (exists M14 M15 M16 o14 o15 o16, M16 = M12 /\ ... /\ ...))))) \/
(exists M5 M6 M7 o5 o6 o7,
 M7 = M4 /\
 join M5 M6 M7 /\
 o7 = o4 /\
 join o5 o6 o7 /\
 (i0 = false /\ M5 = empenv /\ emposabst o5) /\
 (exists x M8 M9 M10 o8 o9 o10,
  M10 = M6 /\
  join M8 M9 M10 /\
  o10 = o6 /\
  join o8 o9 o10 /\
  (gettopis i1 = x /\ M8 = empenv /\ emposabst o8) /\
  ((exists M11 M12 M13 o11 o12 o13,
    M13 = M9 /\
    join M11 M12 M13 /\
    o13 = o9 /\
    join o11 o12 o13 /\
    (x < INUM /\ M11 = empenv /\ emposabst o11) /\
```

```
      (e, e0, M12, i, (i0, i1, c), o12, END None)
      |= invlth_isr I (x + 1) INUM) \/
     x = INUM /\ M9 = empenv /\ emposabst o9))))) /\
(exists M3 M4 M0 o3 o4 o0,
 M0 = M2 /\
 join M3 M4 M0 /\
 o0 = o2 /\
 join o3 o4 o0 /\
 (exists x x0 M5 M6 M7 o5 o6 o7,
  M7 = M3 /\
  join M5 M6 M7 /\
  o7 = o3 /\
  join o5 o6 o7 /\
  (exists b,
   get e OSTCBCur = Some (b, Tptr x) /\
   (x0, Int.unsigned Int.zero) = (b, 0%Z) /\
   M5 = empenv /\ emposabst o5) /\
  mapstoval (x0, Int.unsigned Int.zero) (Tptr x) false (Vptr t) M6 /\
  emposabst o6) /\
 (exists M5 M6 M7 o5 o6 o7,
  M7 = M4 /\
  join M5 M6 M7 /\
  o7 = o4 /\
  join o5 o6 o7 /\
  (e, e0, M5, i, (i0, i1, c), o5, END None) |= pa t lg /\
  (exists M8 M9 M10 o8 o9 o10,
   M10 = M6 /\
   join M8 M9 M10 /\
   o10 = o6 /\
   join o8 o9 o10 /\
   (GoodLInvAsrt pa /\ M8 = empenv /\ emposabst o8) /\
   (exists M11 M12 M13 o11 o12 o13,
    M13 = M9 /\
    join M11 M12 M13 /\
    o13 = o9 /\
    join o11 o12 o13 /\
    (exists M14 M15 M16 o14 o15 o16,
     M16 = M11 /\
     join M14 M15 M16 /\
     o16 = o11 /\
     join o14 o15 o16 /\
     (... /\ ...) /\ (exists M17 M18 M19 o17 o18 o19, ...)) /\
    eq_dom_env e0 nil /\ M12 = empenv /\ emposabst o12))))
```

```
  ============================
   good_is i1

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

mytac.

```
3 subgoals, subgoal 1 (ID 4884)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  e, e0 : env
  m, x, x0 : mem
  x2, x3 : osabst
  x11, x12 : mem
  x14, x15 : osabst
  x5, x6 : mem
  x23 : type
  x61 : block
  H57 : get e OSTCBCur = Some (x61, Tptr x23)
  H20 : GoodLInvAsrt pa
  H26 : eq_dom_env e0 nil
  H3 : forall ab : absop,
       (e, e0, m, empisr, (true, nil, nil), O, ab)
       |= INV I **
          (EX tp : type, GV OSTCBCur @ Tptr tp |-r-> Vptr t) **
          pa t lg **
          [|GoodLInvAsrt pa|] **
          OS [empisr, true, nil, nil] ** A_dom_lenv nil
  H66 : (e, e0, x11, empisr, (true, nil, nil), x14, END None)
        |= getinv (I (S INUM))
  H67 : (exists M1 M2 M o1 o2 o,
         M = x12 /\
         join M1 M2 M /\
         o = x15 /\
         join o1 o2 o /\
         (true = true /\ M1 = empenv /\ emposabst o1) /\
         (exists M3 M4 M0 o3 o4 o0,
          M0 = M2 /\
          join M3 M4 M0 /\
          o0 = o2 /\
          join o3 o4 o0 /\
          (exists x,
           (x 0 = true /\ M3 = empenv /\ emposabst o3) /\
```

```
   empisr = x /\ M3 = empenv /\ emposabst o3 \/
   (exists M5 M6 M7 o5 o6 o7,
    M7 = M3 /\
    join M5 M6 M7 /\
    o7 = o3 /\
    join o5 o6 o7 /\
    ((x 0 = false /\ M5 = empenv /\ emposabst o5) /\
     empisr = x /\ M5 = empenv /\ emposabst o5) /\
    (e, e0, M6, empisr, (true, nil, nil), o6, END None)
    |= getinv (I 0))) /\
  (exists M5 M6 M7 o5 o6 o7,
   M7 = M4 /\
   join M5 M6 M7 /\
   o7 = o4 /\
   join o5 o6 o7 /\
   (exists x,
    (x 1 = true /\ M5 = empenv /\ emposabst o5) /\
    empisr = x /\ M5 = empenv /\ emposabst o5 \/
    (exists M8 M9 M10 o8 o9 o10,
     M10 = M5 /\
     join M8 M9 M10 /\
     o10 = o5 /\
     join o8 o9 o10 /\
     ((x 1 = false /\ M8 = empenv /\ emposabst o8) /\
      empisr = x /\ M8 = empenv /\ emposabst o8) /\
     (e, e0, M9, empisr, (true, nil, nil), o9, END None)
     |= getinv (I 1))) /\
   (exists x,
    (x 2 = true /\ M6 = empenv /\ emposabst o6) /\
    empisr = x /\ M6 = empenv /\ emposabst o6 \/
    (exists M8 M9 M10 o8 o9 o10,
     M10 = M6 /\
     join M8 M9 M10 /\
     o10 = o6 /\
     join o8 o9 o10 /\
     ((x 2 = false /\ M8 = empenv /\ emposabst o8) /\
      empisr = x /\ M8 = empenv /\ emposabst o8) /\
     (e, e0, M9, empisr, (true, nil, nil), o9, END None)
     |= getinv (I 2)))))) \/
(exists M1 M2 M o1 o2 o,
 M = x12 /\
 join M1 M2 M /\
 o = x15 /\
```

```
          join o1 o2 o /\
          (true = false /\ M1 = empenv /\ emposabst o1) /\
          (exists x M3 M4 M0 o3 o4 o0,
           M0 = M2 /\
           join M3 M4 M0 /\
           o0 = o2 /\
           join o3 o4 o0 /\
           (gettopis nil = x /\ M3 = empenv /\ emposabst o3) /\
           ((exists M5 M6 M7 o5 o6 o7,
             M7 = M4 /\
             join M5 M6 M7 /\
             o7 = o4 /\
             join o5 o6 o7 /\
             (x < INUM /\ M5 = empenv /\ emposabst o5) /\
             (e, e0, M6, empisr, (true, nil, nil), o6, END None)
             |= invlth_isr I (x + 1) INUM) \/
            x = INUM /\ M4 = empenv /\ emposabst o4)))
  H7 : join x5 x6 x0
  H65 : join x14 x15 x2
  H63 : join x11 x12 x
  H4 : join x2 x3 O
  H0 : join x x0 m
  H15 : (e, e0, x6, empisr, (true, nil, nil), x3, END None) |= pa t lg
  H58 : mapstoval (x61, Int.unsigned Int.zero) (Tptr x23) false (Vptr t) x5
  ============================
   good_is nil

subgoal 2 (ID 3026) is:
 good_is i1
subgoal 3 (ID 2809) is:
 good_is_S S
```

simpl;auto.

```
2 subgoals, subgoal 1 (ID 3026)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : match
        match get (sig t E) t0 with
        | Some e => Some (G, e, M)
        | None => None
        end
      with
      | Some m =>
          match get (sig t auxs) t0 with
          | Some n => Some (m, isr, n)
          | None => None
          end
      | None => None
      end = Some (e, e0, m, i, (i0, i1, c))
  H0 : t0 <> t
  ============================
   good_is i1

subgoal 2 (ID 2809) is:
 good_is_S S
```

rewrite map_get_sig' in H;auto.

```
2 subgoals, subgoal 1 (ID 4892)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  M : mem
  isr : language.isr
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t : addrval
  lg : list logicvar
  H2 : get O curtid = Some (oscurt t)
  H3 : forall ab : absop, (G, E, M, isr, auxs, O, ab) |= init_cur I pa t lg
  t0 : tid
  e, e0 : env
  m : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H : None = Some (e, e0, m, i, (i0, i1, c))
  H0 : t0 <> t
  ============================
   good_is i1

subgoal 2 (ID 2809) is:
 good_is_S S
```

tryfalse.

```
1 subgoal, subgoal 1 (ID 2809)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  ============================
   good_is_S S
```

unfolds.

```
1 subgoal, subgoal 1 (ID 4907)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  ============================
   forall (t0 : tid) (tst : env * env * mem * language.isr * localst),
   projS S t0 = Some tst ->
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

intros.

```
1 subgoal, subgoal 1 (ID 4910)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  t0 : tid
  tst : env * env * mem * language.isr * localst
  H7 : projS S t0 = Some tst
  ============================
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

assert (t0 = t / t0 <> t) by tauto.

```
H8 : t0 = t \/ t0 <> t
```

destruct H8.

```
2 subgoals, subgoal 1 (ID 5218)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  t0 : tid
  tst : env * env * mem * language.isr * localst
  H7 : projS S t0 = Some tst
  H8 : t0 = t
  ============================
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
```

```
  let (p1, _) := p0 in
  let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

subst.

```
2 subgoals, subgoal 1 (ID 5229)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  tst : env * env * mem * language.isr * localst
  H7 : projS (G, envs, M, isr, lst) t = Some tst
  ============================
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

destruct tst.

```
2 subgoals, subgoal 1 (ID 5237)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  p : env * env * mem * language.isr
  l : localst
  H7 : projS (G, envs, M, isr, lst) t = Some (p, l)
  ============================
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

destruct p.

destruct p.

destruct p.

destruct l.

destruct p.

```
2 subgoals, subgoal 1 (ID 5285)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS (G, envs, M, isr, lst) t = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

unfolds in H7.

```
2 subgoals, subgoal 1 (ID 5286)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match projD (G, envs, M) t with
       | Some m =>
           match get lst t with
           | Some n => Some (m, isr, n)
           | None => None
           end
       | None => None
       end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1
```

```
subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

unfold projD in H7.

```
2 subgoals, subgoal 1 (ID 5287)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match
         match get envs t with
         | Some e => Some (G, e, M)
         | None => None
         end
       with
       | Some m =>
           match get lst t with
           | Some n => Some (m, isr, n)
           | None => None
           end
```

```
        | None => None
        end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

eapply join_sig_get_disj in H0;eauto.

```
2 subgoals, subgoal 1 (ID 5289)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : get envs t = Some E /\ get envs' t = None /\ Maps.sub envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match
         match get envs t with
         | Some e => Some (G, e, M)
         | None => None
         end
       with
       | Some m =>
           match get lst t with
           | Some n => Some (m, isr, n)
           | None => None
           end
```

```
        | None => None
        end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

destruct H0.

```
2 subgoals, subgoal 1 (ID 5301)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : get envs t = Some E
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match
         match get envs t with
         | Some e => Some (G, e, M)
         | None => None
         end
       with
       | Some m =>
           match get lst t with
           | Some n => Some (m, isr, n)
           | None => None
```

```
            end
          | None => None
          end = Some (e, e0, m0, i, (i0, i1, c))
    ============================
     good_is i1

  subgoal 2 (ID 5219) is:
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

eapply join_sig_get_disj in H1;eauto.

```
2 subgoals, subgoal 1 (ID 5303)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : get envs t = Some E
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H1 : get lst t = Some auxs /\ get lst' t = None /\ Maps.sub lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match
         match get envs t with
         | Some e => Some (G, e, M)
         | None => None
         end
       with
       | Some m =>
           match get lst t with
           | Some n => Some (m, isr, n)
           | None => None
```

```
            end
          | None => None
          end = Some (e, e0, m0, i, (i0, i1, c))
    ============================
     good_is i1

  subgoal 2 (ID 5219) is:
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

unfold tid in *.

```
2 subgoals, subgoal 1 (ID 5345)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : addrval -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : get envs t = Some E
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H1 : get lst t = Some auxs /\ get lst' t = None /\ Maps.sub lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match
         match get envs t with
         | Some e => Some (G, e, M)
         | None => None
         end
       with
       | Some m =>
           match get lst t with
           | Some n => Some (m, isr, n)
           | None => None
```

```
            end
          | None => None
          end = Some (e, e0, m0, i, (i0, i1, c))
    ============================
     good_is i1

  subgoal 2 (ID 5219) is:
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

rewrite H in H7.

```
2 subgoals, subgoal 1 (ID 5347)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : addrval -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : get envs t = Some E
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H1 : get lst t = Some auxs /\ get lst' t = None /\ Maps.sub lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match get lst t with
       | Some n => Some (G, E, M, isr, n)
       | None => None
       end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
```

```
  let (p1, _) := p0 in
  let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

destruct H1.

```
2 subgoals, subgoal 1 (ID 5354)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : addrval -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : get envs t = Some E
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H1 : get lst t = Some auxs
  H8 : get lst' t = None /\ Maps.sub lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : match get lst t with
       | Some n => Some (G, E, M, isr, n)
       | None => None
       end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
```

```
  let (p0, _) := p in
  let (p1, _) := p0 in
  let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

rewrite H1 in H7.

```
2 subgoals, subgoal 1 (ID 5356)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : addrval -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : get envs t = Some E
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H1 : get lst t = Some auxs
  H8 : get lst' t = None /\ Maps.sub lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : Some (G, E, M, isr, auxs) = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
```

```
  let (p1, _) := p0 in
  let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

inverts H7.

```
2 subgoals, subgoal 1 (ID 5500)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  envs, envs' : cltenvs
  m, M' : mem
  lst, lst' : ltaskstset
  I : Inv
  pa : addrval -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : get envs' t = None /\ Maps.sub envs' envs
  H8 : get lst' t = None /\ Maps.sub lst' lst
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  e, e0 : env
  m0 : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H : get envs t = Some e0
  H2 : join m M' m0
  H6 : initst (e, envs', M', i, lst') O I pa lg
  IHinitst : good_is_S (e, envs', M', i, lst')
  H1 : get lst t = Some (i0, i1, c)
  H5 : forall ab : absop,
       (e, e0, m, i, (i0, i1, c), empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  ============================
   good_is i1

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

lets Hx: H5 (spec_done None).

```
    Hx : (e, e0, m, i, (i0, i1, c), empenv, END None)
         |= init_rdy pa t lg ** A_dom_lenv ni
```

simpl in Hx;mytac;auto.

```
2 subgoals, subgoal 1 (ID 6601)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  envs, envs' : cltenvs
  m, M' : mem
  lst, lst' : ltaskstset
  I : Inv
  pa : addrval -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : get envs' t = None
  H59 : Maps.sub envs' envs
  H8 : get lst' t = None
  H58 : Maps.sub lst' lst
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  e, e0 : env
  m0 : mem
  H : get envs t = Some e0
  H2 : join m M' m0
  H24 : GoodLInvAsrt pa
  H30, H13 : eq_dom_env e0 nil
  H1 : get lst t = Some (true, nil, nil)
  H6 : initst (e, envs', M', empisr, lst') O I pa lg
  IHinitst : good_is_S (e, envs', M', empisr, lst')
  H5 : forall ab : absop,
       (e, e0, m, empisr, (true, nil, nil), empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H19 : (e, e0, m, empisr, (true, nil, nil), empenv, END None) |= pa t lg
  ============================
   good_is nil

subgoal 2 (ID 5219) is:
 let (p, l) := tst in
 let (p0, _) := p in
 let (p1, _) := p0 in
 let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

simpl;auto.

```
1 subgoal, subgoal 1 (ID 5219)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  t0 : tid
  tst : env * env * mem * language.isr * localst
  H7 : projS S t0 = Some tst
  H8 : t0 <> t
  ============================
   let (p, l) := tst in
   let (p0, _) := p in
   let (p1, _) := p0 in
   let (_, _) := p1 in let (p2, _) := l in let (_, f) := p2 in good_is f
```

destruct tst.

destruct p.

destruct p.

destruct p.

destruct l.
destruct p.

```
1 subgoal, subgoal 1 (ID 6669)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : good_is_S (G, envs', M', isr, lst')
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
  H8 : t0 <> t
  ============================
   good_is i1
```

unfolds in IHinitst.

```
1 subgoal, subgoal 1 (ID 6670)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
```

```
H8 : t0 <> t
============================
 good_is i1
```

assert (projS (G, envs', M', isr, lst') t0 = Some (e, e0, M', i, (i0, i1, c))).

```
2 subgoals, subgoal 1 (ID 6693)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
```

```
  H8 : t0 <> t
  ============================
   projS (G, envs', M', isr, lst') t0 = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

unfolds.

```
2 subgoals, subgoal 1 (ID 6695)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
```

```
  H8 : t0 <> t
  ============================
   match projD (G, envs', M') t0 with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

unfold projD.

```
2 subgoals, subgoal 1 (ID 6696)

 sd : ossched
 init : osstate -> osabst -> Type
 S : env * cltenvs * mem * language.isr * ltaskstset
 O : osabst
 G : env
 envs, envs' : cltenvs
 M, m, M' : mem
 isr : language.isr
 lst, lst' : ltaskstset
 E : env
 auxs : localst
 I : Inv
 pa : tid -> list logicvar -> asrt
 t, tc : addrval
 lg : list logicvar
 H : S = (G, envs, M, isr, lst)
 H0 : join (sig t E) envs' envs
 H1 : join (sig t auxs) lst' lst
 H2 : join m M' M
 H3 : get O curtid = Some (oscurt tc)
 H4 : t <> tc
 H5 : forall ab : absop,
      (G, E, m, isr, auxs, empenv, ab)
      |= init_rdy pa t lg ** A_dom_lenv nil
 H6 : initst (G, envs', M', isr, lst') O I pa lg
 IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
            projS (G, envs', M', isr, lst') t = Some tst ->
            let (p, l) := tst in
            let (p0, _) := p in
            let (p1, _) := p0 in
            let (_, _) := p1 in
            let (p2, _) := l in let (_, f) := p2 in good_is f
 t0 : tid
 e, e0 : env
 m0 : mem
 i : language.isr
 i0 : ie
 i1 : is
 c : cs
 H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
```

```
  H8 : t0 <> t
  ============================
   match
     match get envs' t0 with
     | Some e1 => Some (G, e1, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
good_is i1
```

unfolds in H7.

```
2 subgoals, subgoal 1 (ID 6697)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : (let (p, z) := S in
```

```
        let (x, y) := p in
        match projD x t0 with
        | Some m =>
            match get z t0 with
            | Some n => Some (m, y, n)
            | None => None
            end
        | None => None
        end) = Some (e, e0, m0, i, (i0, i1, c))
  H8 : t0 <> t
  ============================
   match
     match get envs' t0 with
     | Some e1 => Some (G, e1, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

destruct S.

```
2 subgoals, subgoal 1 (ID 6711)

  sd : ossched
  init : osstate -> osabst -> Type
  p : env * cltenvs * mem * language.isr
  l : ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : (p, l) = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
```

```
  H7 : (let (x, y) := p in
        match projD x t0 with
        | Some m =>
            match get l t0 with
            | Some n => Some (m, y, n)
            | None => None
            end
        | None => None
        end) = Some (e, e0, m0, i, (i0, i1, c))
  H8 : t0 <> t
  ============================
   match
     match get envs' t0 with
     | Some e1 => Some (G, e1, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

destruct p.

```
2 subgoals, subgoal 1 (ID 6725)

  sd : ossched
  init : osstate -> osabst -> Type
  p : env * cltenvs * mem
  i2 : language.isr
  l : ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : (p, i2, l) = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
```

```
  c : cs
  H7 : match projD p t0 with
       | Some m =>
           match get l t0 with
           | Some n => Some (m, i2, n)
           | None => None
           end
       | None => None
       end = Some (e, e0, m0, i, (i0, i1, c))
  H8 : t0 <> t
  ============================
   match
     match get envs' t0 with
     | Some e1 => Some (G, e1, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

inverts H.

```
2 subgoals, subgoal 1 (ID 6833)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  H7 : match projD (G, envs, M) t0 with
       | Some m =>
```

```
          match get lst t0 with
          | Some n => Some (m, isr, n)
          | None => None
          end
      | None => None
      end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   match
     match get envs' t0 with
     | Some e1 => Some (G, e1, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

unfold projD in H7.

```
2 subgoals, subgoal 1 (ID 6834)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  H7 : match
         match get envs t0 with
```

```
          | Some e => Some (G, e, M)
          | None => None
          end
        with
        | Some m =>
            match get lst t0 with
            | Some n => Some (m, isr, n)
            | None => None
            end
        | None => None
        end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   match
     match get envs' t0 with
     | Some e1 => Some (G, e1, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

remember (get envs t0) as X.

```
lg : list logicvar
H0 : join (sig t E) envs' envs
H1 : join (sig t auxs) lst' lst
H2 : join m M' M
H3 : get O curtid = Some (oscurt tc)
H4 : t <> tc
H5 : forall ab : absop,
     (G, E, m, isr, auxs, empenv, ab)
     |= init_rdy pa t lg ** A_dom_lenv nil
H6 : initst (G, envs', M', isr, lst') O I pa lg
IHinitst : forall (t : tid)
             (tst : env * env * mem * language.isr * localst),
           projS (G, envs', M', isr, lst') t = Some tst ->
           let (p, l) := tst in
           let (p0, _) := p in
           let (p1, _) := p0 in
           let (_, _) := p1 in
           let (p2, _) := l in let (_, f) := p2 in good_is f
t0 : tid
e, e0 : env
m0 : mem
i : language.isr
i0 : ie
i1 : is
c : cs
H8 : t0 <> t
X : option env
HeqX : X = get envs t0
H7 : match match X with
           | Some e => Some (G, e, M)
           | None => None
           end with
     | Some m =>
         match get lst t0 with
         | Some n => Some (m, isr, n)
         | None => None
         end
     | None => None
     end = Some (e, e0, m0, i, (i0, i1, c))
============================
 match
   match get envs' t0 with
   | Some e1 => Some (G, e1, M')
```

```
      | None => None
      end
    with
    | Some m1 =>
        match get lst' t0 with
        | Some n => Some (m1, isr, n)
        | None => None
        end
    | None => None
    end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

destruct X;tryfalse.

```
sd : ossched
 init : osstate -> osabst -> Type
 O : osabst
 G : env
 envs, envs' : cltenvs
 M, m, M' : mem
 isr : language.isr
 lst, lst' : ltaskstset
 E : env
 auxs : localst
 I : Inv
 pa : tid -> list logicvar -> asrt
 t, tc : addrval
 lg : list logicvar
 H0 : join (sig t E) envs' envs
 H1 : join (sig t auxs) lst' lst
 H2 : join m M' M
 H3 : get O curtid = Some (oscurt tc)
 H4 : t <> tc
 H5 : forall ab : absop,
      (G, E, m, isr, auxs, empenv, ab)
      |= init_rdy pa t lg ** A_dom_lenv nil
 H6 : initst (G, envs', M', isr, lst') O I pa lg
 IHinitst : forall (t : tid)
              (tst : env * env * mem * language.isr * localst),
            projS (G, envs', M', isr, lst') t = Some tst ->
            let (p, l) := tst in
            let (p0, _) := p in
            let (p1, _) := p0 in
            let (_, _) := p1 in
            let (p2, _) := l in let (_, f) := p2 in good_is f
 t0 : tid
 e, e0 : env
 m0 : mem
 i : language.isr
 i0 : ie
 i1 : is
 c : cs
 H8 : t0 <> t
 e1 : env
 HeqX : Some e1 = get envs t0
 H7 : match get lst t0 with
      | Some n => Some (G, e1, M, isr, n)
```

```
        | None => None
        end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   match
     match get envs' t0 with
     | Some e2 => Some (G, e2, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

remember (get lst t0) as Y.

```
2 subgoals, subgoal 1 (ID 6888)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
```

```
  Y : option localst
  HeqY : Y = get lst t0
  H7 : match Y with
       | Some n => Some (G, e1, M, isr, n)
       | None => None
       end = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   match
     match get envs' t0 with
     | Some e2 => Some (G, e2, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

destruct Y;tryfalse.

```
2 subgoals, subgoal 1 (ID 6901)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
```

```
  l : localst
  HeqY : Some l = get lst t0
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   match
     match get envs' t0 with
     | Some e2 => Some (G, e2, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6694) is:
 good_is i1
```

eapply join_get_get_r_rev with (a:=t0) in H0;eauto.

```
3 subgoals, subgoal 1 (ID 6924)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
  l : localst
```

```
  HeqY : Some l = get lst t0
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  H0 : get envs' t0 = Some e1
  ============================
   match
     match get envs' t0 with
     | Some e2 => Some (G, e2, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6927) is:
 get (sig t E) t0 = None
subgoal 3 (ID 6694) is:
 good_is i1
```

eapply join_get_get_r_rev with (a:=t0) in H1;eauto.

```
4 subgoals, subgoal 1 (ID 6938)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
  l : localst
  HeqY : Some l = get lst t0
```

```
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  H0 : get envs' t0 = Some e1
  H1 : get lst' t0 = Some l
  ============================
   match
     match get envs' t0 with
     | Some e2 => Some (G, e2, M')
     | None => None
     end
   with
   | Some m1 =>
       match get lst' t0 with
       | Some n => Some (m1, isr, n)
       | None => None
       end
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6941) is:
 get (sig t auxs) t0 = None
subgoal 3 (ID 6927) is:
 get (sig t E) t0 = None
subgoal 4 (ID 6694) is:
 good_is i1
```

rewrite H0.

```
4 subgoals, subgoal 1 (ID 6949)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
  l : localst
  HeqY : Some l = get lst t0
```

```
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  H0 : get envs' t0 = Some e1
  H1 : get lst' t0 = Some l
  ============================
   match get lst' t0 with
   | Some n => Some (G, e1, M', isr, n)
   | None => None
   end = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6941) is:
 get (sig t auxs) t0 = None
subgoal 3 (ID 6927) is:
 get (sig t E) t0 = None
subgoal 4 (ID 6694) is:
 good_is i1
```

rewrite H1.

```
4 subgoals, subgoal 1 (ID 6950)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
  l : localst
  HeqY : Some l = get lst t0
```

```
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  H0 : get envs' t0 = Some e1
  H1 : get lst' t0 = Some l
  ============================
   Some (G, e1, M', isr, l) = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6941) is:
 get (sig t auxs) t0 = None
subgoal 3 (ID 6927) is:
 get (sig t E) t0 = None
subgoal 4 (ID 6694) is:
 good_is i1
```

inverts H7.

```
4 subgoals, subgoal 1 (ID 7098)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  envs, envs' : cltenvs
  m, M' : mem
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  t0 : tid
  e, e0 : env
  m0 : mem
  i : isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  HeqX : Some e0 = get envs t0
  H0 : get envs' t0 = Some e0
  H2 : join m M' m0
  H5 : forall ab : absop,
       (e, E, m, i, auxs, empenv, ab) |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (e, envs', M', i, lst') O I pa lg
  IHinitst : forall (t : tid) (tst : env * env * mem * isr * localst),
             projS (e, envs', M', i, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  HeqY : Some (i0, i1, c) = get lst t0
  H1 : get lst' t0 = Some (i0, i1, c)
  ============================
   Some (e, e0, M', i, (i0, i1, c)) = Some (e, e0, M', i, (i0, i1, c))

subgoal 2 (ID 6941) is:
```

```
  get (sig t auxs) t0 = None
subgoal 3 (ID 6927) is:
 get (sig t E) t0 = None
subgoal 4 (ID 6694) is:
 good_is i1
```

auto.

```
3 subgoals, subgoal 1 (ID 6941)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
  l : localst
```

```
  HeqY : Some l = get lst t0
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  H0 : get envs' t0 = Some e1
  ============================
   get (sig t auxs) t0 = None

subgoal 2 (ID 6927) is:
 get (sig t E) t0 = None
subgoal 3 (ID 6694) is:
 good_is i1
```

eapply map_get_sig';eauto.

```
2 subgoals, subgoal 1 (ID 6927)

  sd : ossched
  init : osstate -> osabst -> Type
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H8 : t0 <> t
  e1 : env
  HeqX : Some e1 = get envs t0
```

```
  l : localst
  HeqY : Some l = get lst t0
  H7 : Some (G, e1, M, isr, l) = Some (e, e0, m0, i, (i0, i1, c))
  ============================
   get (sig t E) t0 = None

subgoal 2 (ID 6694) is:
 good_is i1
```

eapply map_get_sig';eauto.

```
1 subgoal, subgoal 1 (ID 6694)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
```

```
H8 : t0 <> t
H9 : projS (G, envs', M', isr, lst') t0 = Some (e, e0, M', i, (i0, i1, c))
============================
 good_is i1
```

apply IHinitst in H9.

```
1 subgoal, subgoal 1 (ID 7141)

  sd : ossched
  init : osstate -> osabst -> Type
  S : env * cltenvs * mem * language.isr * ltaskstset
  O : osabst
  G : env
  envs, envs' : cltenvs
  M, m, M' : mem
  isr : language.isr
  lst, lst' : ltaskstset
  E : env
  auxs : localst
  I : Inv
  pa : tid -> list logicvar -> asrt
  t, tc : addrval
  lg : list logicvar
  H : S = (G, envs, M, isr, lst)
  H0 : join (sig t E) envs' envs
  H1 : join (sig t auxs) lst' lst
  H2 : join m M' M
  H3 : get O curtid = Some (oscurt tc)
  H4 : t <> tc
  H5 : forall ab : absop,
       (G, E, m, isr, auxs, empenv, ab)
       |= init_rdy pa t lg ** A_dom_lenv nil
  H6 : initst (G, envs', M', isr, lst') O I pa lg
  IHinitst : forall (t : tid)
               (tst : env * env * mem * language.isr * localst),
             projS (G, envs', M', isr, lst') t = Some tst ->
             let (p, l) := tst in
             let (p0, _) := p in
             let (p1, _) := p0 in
             let (_, _) := p1 in
             let (p2, _) := l in let (_, f) := p2 in good_is f
  t0 : tid
  e, e0 : env
  m0 : mem
  i : language.isr
  i0 : ie
  i1 : is
  c : cs
  H7 : projS S t0 = Some (e, e0, m0, i, (i0, i1, c))
```

```
H8 : t0 <> t
H9 : good_is i1
============================
 good_is i1
```

auto.
Qed.