EVENT: Start with the library `"c-begin"`.

EVENT: Enable clock-predefined-proc-call-sequence.

EVENT: Enable clock-predefined-proc-call-body-translation.

EVENT: Enable predefined-proc-call-sequence.

EVENT: Disable mg-to-p-simple-literal.

EVENT: Disable ilessp.

EVENT: Disable not-bool.

EVENT: Disable inegate.

EVENT: Disable fix-small-integer.

EVENT: Disable iplus.

EVENT: Disable idifference.

EVENT: Disable signatures-match-preserves-plistp.

THEOREM: predefined-proc-call-meaning-r-2
(car (*stmt*) = `'predefined-proc-call-mg`)
→ (mg-meaning-r (*stmt*, *proc-list*, *mg-state*, *n*, *sizes*)
= **if** *n* ≃ `0` **then** signal-system-error (*mg-state*, `'timed-out`)
elseif ¬ normal (*mg-state*) **then** *mg-state*
elseif resources-inadequatep (*stmt*, *proc-list*, *sizes*)
then signal-system-error (*mg-state*, `'resource-error`)
else mg-meaning-predefined-proc-call (*stmt*, *mg-state*) **endif**)

THEOREM: predefined-call-translation-2
(car (*stmt*) = `'predefined-proc-call-mg`)
→ (translate (*cinfo*, *cond-list*, *stmt*, *proc-list*)
= add-code (*cinfo*,
predefined-proc-call-sequence (*stmt*,

label-alist (*cinfo*))))

THEOREM: unlabel-call
unlabel (cons (`'call`, x)) = cons (`'call`, x)

THEOREM: lessp-add1-add1-2
((1 + (1 + x)) < `2`) = **f**

THEOREM: lessp-add1-add1-add1-3
((1 + (1 + (1 + x))) < `3`) = **f**

```
;; >> get rid of the extra "d"
```

THEOREM: simple-typed-literalp-mapping-listp
simple-typed-literalp (*lit*, *type*) → listp (mg-to-p-simple-literal (*lit*))

THEOREM: boolean-mg-to-p-simple-literal
(x ∈ `'(true-mg false-mg)`)
→ (mg-to-p-simple-literal (tag (`'boolean-mg`, x))
= tag (`'bool`,
if x = `'false-mg` **then** `'f`
else `'t` **endif**))

THEOREM: boolean-mg-to-p-simple-literal2
mg-to-p-simple-literal (mg-bool (x))
= tag (`'bool`,
if ¬ x **then** `'f`
else `'t` **endif**)

THEOREM: mg-to-p-simple-literalp-preserves-untag-equality
(simple-typed-literalp (x, *type*) ∧ simple-typed-literalp (y, *type*))
→ ((untag (mg-to-p-simple-literal (x))
= untag (mg-to-p-simple-literal (y)))
= (untag (x) = untag (y)))

THEOREM: mg-to-p-simple-literalp-preserves-untag-ilessp
(int-literalp (x) ∧ int-literalp (y))
→ (ilessp (untag (mg-to-p-simple-literal (x)),
untag (mg-to-p-simple-literal (y)))
= ilessp (untag (x), untag (y)))

THEOREM: simple-identifiers-have-simple-values
(mg-alistp (*alist*) ∧ simple-identifierp (x, *alist*))
→ simple-typed-literalp (caddr (assoc (x, *alist*)), cadr (assoc (x, *alist*)))

THEOREM: alist-element-type-conversion
(mg-alistp (*lst*) ∧ simple-identifierp (*x*, *lst*))
→ (type (mg-to-p-simple-literal (caddr (assoc (*x*, *lst*))))
= **if** cadr (assoc (*x*, *lst*)) = `'boolean-mg` **then** `'bool`
else `'int` **endif**)

THEOREM: literal-type-conversion
simple-typed-literalp (*lit*, *type*)
→ (type (mg-to-p-simple-literal (*lit*))
= **if** *type* = `'boolean-mg` **then** `'bool`
else `'int` **endif**)

THEOREM: int-literals-mapping
int-literalp (*x*)
→ ((type (mg-to-p-simple-literal (*x*)) = `'int`)
∧ listp (mg-to-p-simple-literal (*x*))
∧ (cddr (mg-to-p-simple-literal (*x*)) = **nil**)
∧ small-integerp (untag (mg-to-p-simple-literal (*x*)), 32))

THEOREM: int-identifiers-have-int-literal-values
(mg-alistp (*mg-vars*) ∧ int-identifierp (*x*, *mg-vars*))
→ int-literalp (caddr (assoc (*x*, *mg-vars*)))

THEOREM: boolean-identifiers-have-boolean-literal-values
(mg-alistp (*mg-vars*) ∧ boolean-identifierp (*x*, *mg-vars*))
→ boolean-literalp (caddr (assoc (*x*, *mg-vars*)))

THEOREM: length-plistp-2-rewrite
(cddr (*x*) = **nil**) = length-plistp (*x*, 2)

EVENT: Disable length-plistp-2-rewrite.

```
;; There is no reason why this couldn't be changed to defined-identifierp's
;; rather than simple-identifierp's
```

THEOREM: simple-identifier-mapping
(mg-vars-list-ok-in-p-state (*mg-vars*, *bindings*, *temp-stk*)
∧ simple-identifierp (*b*, *mg-vars*)
∧ (length (*temp-stk*) < MG-MAX-TEMP-STK-SIZE)
∧ all-cars-unique (*mg-vars*))
→ ((type (value (*b*, *bindings*)) = `'nat`)
∧ (cddr (value (*b*, *bindings*)) = **nil**)
∧ listp (value (*b*, *bindings*))
∧ small-naturalp (untag (value (*b*, *bindings*)), 32))

```
;; I think this is probably redundant.
```

THEOREM: simple-identifier-mapping-2
(mg-vars-list-ok-in-p-state (mg-alist (*mg-state*), *bindings*, *temp-stk*)
 ∧ simple-identifierp (*b*, mg-alist (*mg-state*))
 ∧ (length (*temp-stk*) < MG-MAX-TEMP-STK-SIZE)
 ∧ all-cars-unique (mg-alist (*mg-state*)))
→ ((type (value (*b*, *bindings*)) = 'nat
  ∧ (cddr (value (*b*, *bindings*)) = **nil**)
  ∧ listp (value (*b*, *bindings*))
  ∧ small-naturalp (untag (value (*b*, *bindings*)), 32))

THEOREM: simple-identifier-mapping-3
(mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*)
 ∧ definedp (*b*, *mg-alist*))
→ ((untag (value (*b*, *bindings*)) < (1 + length (*temp-stk*))) = **t**)

THEOREM: simple-identifier-nat-p-objectp
(mg-vars-list-ok-in-p-state (mg-alist (*mg-state*), *bindings*, *temp-stk*)
 ∧ simple-identifierp (*b*, mg-alist (*mg-state*))
 ∧ (length (*temp-stk*) < MG-MAX-TEMP-STK-SIZE)
 ∧ all-cars-unique (mg-alist (*mg-state*))
 ∧ (p-word-size (*state*) = MG-WORD-SIZE))
→ p-objectp-type ('nat, value (*b*, *bindings*), *state*)

EVENT: Enable mg-var-ok-in-p-state.

THEOREM: array-identifier-nat-p-objectp
(mg-vars-list-ok-in-p-state (*mg-alist*, *bindings*, *temp-stk*)
 ∧ definedp (*b*, *mg-alist*)
 ∧ (length (*temp-stk*) < MG-MAX-TEMP-STK-SIZE)
 ∧ (p-word-size (*state*) = MG-WORD-SIZE))
→ p-objectp-type ('nat, value (*b*, *bindings*), *state*)

THEOREM: int-literalp-mapping
int-literalp (*x*) → (untag (mg-to-p-simple-literal (*x*)) = untag (*x*))

THEOREM: boolean-literalp-mapping
boolean-literalp (*x*)
→ (untag (mg-to-p-simple-literal (*x*))
  = **if** untag (*x*) = 'false-mg **then** 'f
  else 't **endif**)

THEOREM: untag-boolean
boolean-literalp (*x*) → (untag (*x*) ∈ '(true-mg false-mg))

THEOREM: int-literalp-value-small
int-literalp $(x) \rightarrow$ small-integerp (untag (x), `32`)

THEOREM: small-integerp-mapping
small-integerp (x, MG-WORD-SIZE)
$\rightarrow$ (mg-to-p-simple-literal (tag (`'int-mg`, x)) = tag (`'int`, x))

THEOREM: special-conditions-mg-cond-to-p-nat
(mg-cond-to-p-nat (`'routineerror`, *state*) = `'(nat 1)`)
$\wedge$ (mg-cond-to-p-nat (`'normal`, *state*) = `'(nat 2)`)

THEOREM: fix-small-integer-identity
small-integerp $(x, n) \rightarrow$ (fix-small-integer $(x, n) = x$)

THEOREM: int-literal-int-objectp
(int-literalp $(x) \wedge$ (p-word-size (*state*) = MG-WORD-SIZE))
$\rightarrow$ p-objectp-type (`'int`, mg-to-p-simple-literal (x), *state*)

THEOREM: untag-int-literal-integerp
int-literalp $(x) \rightarrow$ integerp (untag (x))

THEOREM: bool-literal-bool-objectp
p-objectp-type (`'bool`, `'(bool t)`, *state*)
$\wedge$ p-objectp-type (`'bool`, `'(bool f)`, *state*)

THEOREM: bool-literal-bool-objectp2
boolean-literalp (x)
$\rightarrow$ p-objectp-type (`'bool`, mg-to-p-simple-literal (x), *state*)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                            ;;
;;               EXACT-TIME LEMMA MG-SIMPLE-VARIABLE-ASSIGNMENT               ;;
;;                                                                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: mg-simple-variable-assignment-args-definedp
(ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ (car (*stmt*) = `'predefined-proc-call-mg`)
$\wedge$ (call-name (*stmt*) = `'mg-simple-variable-assignment`)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*))
$\rightarrow$ (definedp (car (call-actuals (*stmt*)), mg-alist (*mg-state*))
$\wedge$ definedp (cadr (call-actuals (*stmt*)), mg-alist (*mg-state*)))

THEOREM: mg-simple-variable-assignment-args-simple-identifierps
(ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 ∧ (car (*stmt*) = `'predefined-proc-call-mg`)
 ∧ (call-name (*stmt*) = `'mg-simple-variable-assignment`)
 ∧ ok-mg-statep (*mg-state*, *r-cond-list*)
 ∧ signatures-match (mg-alist (*mg-state*), *name-alist*))
→ (simple-identifierp (car (call-actuals (*stmt*)), mg-alist (*mg-state*))
  ∧ simple-identifierp (cadr (call-actuals (*stmt*)),
  mg-alist (*mg-state*)))

THEOREM: mg-simple-variable-assignment-steps-1-2
((¬ resources-inadequatep (*stmt*,
  proc-list,
  list (length (*temp-stk*), p-ctrl-stk-size (*ctrl-stk*))))
 ∧ (car (*stmt*) = `'predefined-proc-call-mg`)
 ∧ (call-name (*stmt*) = `'mg-simple-variable-assignment`)
 ∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 ∧ ok-mg-def-plistp (*proc-list*)
 ∧ ok-mg-statep (*mg-state*, *r-cond-list*)
 ∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
  = append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
  code2))
 ∧ user-defined-procp (*subr*, *proc-list*)
 ∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
  bindings (top (*ctrl-stk*)),
  temp-stk)
 ∧ normal (*mg-state*))
→ (p-step (p-step (map-down (*mg-state*,
  proc-list,
  ctrl-stk,
  temp-stk,
  tag (`'pc`, cons (*subr*, length (code (*cinfo*)))),
  t-cond-list)))
  = p-state (tag (`'pc`, cons (*subr*, length (code (*cinfo*)) + `2`)),
  ctrl-stk,
  push (value (cadr (call-actuals (*stmt*)),
  bindings (top (*ctrl-stk*))),
  push (value (car (call-actuals (*stmt*)),
  bindings (top (*ctrl-stk*))),
  map-down-values (mg-alist (*mg-state*),
  bindings (top (*ctrl-stk*)),
  temp-stk))),
  translate-proc-list (*proc-list*),
  list (list (`'c-c`,

```
                    mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))

;; (call mg-simple-variable-assignment)
```

THEOREM: mg-simple-variable-assignment-step-3

```
((¬ resources-inadequatep (stmt,
                           proc-list,
                           list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'predefined-proc-call-mg)
 ∧  (call-name (stmt) = 'mg-simple-variable-assignment)
 ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧  ok-mg-def-plistp (proc-list)
 ∧  ok-mg-statep (mg-state, r-cond-list)
 ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧  user-defined-procp (subr, proc-list)
 ∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 ∧  normal (mg-state))
→   (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 2)),
                      ctrl-stk,
                      push (value (cadr (call-actuals (stmt)),
                                  bindings (top (ctrl-stk))),
                            push (value (car (call-actuals (stmt)),
                                        bindings (top (ctrl-stk))),
                                  map-down-values (mg-alist (mg-state),
                                                   bindings (top (ctrl-stk)),
                                                   temp-stk))),
                      translate-proc-list (proc-list),
                      list (list ('c-c,
                                  mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                      MG-MAX-CTRL-STK-SIZE,
                      MG-MAX-TEMP-STK-SIZE,
                      MG-WORD-SIZE,
                      'run))
    =  p-state (tag ('pc,
                     '(mg-simple-variable-assignment . 0)),
```

```
                push (p-frame (list (cons ('dest,
                                           value (car (call-actuals (stmt)),
                                                  bindings (top (ctrl-stk)))),
                                    cons ('source,
                                          value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk))))),
                              tag ('pc,
                                   cons (subr, length (code (cinfo))
                                               +  3))),
                      ctrl-stk),
                map-down-values (mg-alist (mg-state),
                                 bindings (top (ctrl-stk)),
                                 temp-stk),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))

;; (push-local source)
```

THEOREM: mg-simple-variable-assignment-step-4

```
((¬ resources-inadequatep (stmt,
                           proc-list,
                           list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'predefined-proc-call-mg)
 ∧  (call-name (stmt) = 'mg-simple-variable-assignment)
 ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧  ok-mg-def-plistp (proc-list)
 ∧  ok-mg-statep (mg-state, r-cond-list)
 ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧  user-defined-procp (subr, proc-list)
 ∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 ∧  normal (mg-state))
 →  (p-step (p-state (tag ('pc,
                           '(mg-simple-variable-assignment . 0)),
                     push (p-frame (list (cons ('dest,
```

```
                                        value (car (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                    cons ('source,
                                        value (cadr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk))))),
                                tag ('pc,
                                    cons (subr, length (code (cinfo))
                                                + 3))),
                            ctrl-stk),
                        map-down-values (mg-alist (mg-state),
                                         bindings (top (ctrl-stk)),
                                         temp-stk),
                        translate-proc-list (proc-list),
                        list (list ('c-c,
                                    mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                        MG-MAX-CTRL-STK-SIZE,
                        MG-MAX-TEMP-STK-SIZE,
                        MG-WORD-SIZE,
                        'run))
=   p-state (tag ('pc,
                  '(mg-simple-variable-assignment . 1)),
             push (p-frame (list (cons ('dest,
                                        value (car (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                    cons ('source,
                                        value (cadr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk))))),
                                tag ('pc,
                                    cons (subr, length (code (cinfo))
                                                + 3))),
                            ctrl-stk),
             push (value (cadr (call-actuals (stmt)),
                               bindings (top (ctrl-stk))),
                         map-down-values (mg-alist (mg-state),
                                          bindings (top (ctrl-stk)),
                                          temp-stk)),
             translate-proc-list (proc-list),
             list (list ('c-c,
                         mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
             MG-MAX-CTRL-STK-SIZE,
             MG-MAX-TEMP-STK-SIZE,
             MG-WORD-SIZE,
             'run))
```

THEOREM: mg-simple-variable-assignment-step-5

```
((¬ resources-inadequatep (stmt,
                           proc-list,
                           list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'predefined-proc-call-mg)
 ∧  (call-name (stmt) = 'mg-simple-variable-assignment)
 ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧  ok-mg-def-plistp (proc-list)
 ∧  ok-mg-statep (mg-state, r-cond-list)
 ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧  user-defined-procp (subr, proc-list)
 ∧  all-cars-unique (mg-alist (mg-state))
 ∧  signatures-match (mg-alist (mg-state), name-alist)
 ∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 ∧  normal (mg-state))
→  (p-step (p-state (tag ('pc,
                          '(mg-simple-variable-assignment . 1)),
                    push (p-frame (list (cons ('dest,
                                               value (car (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk)))),
                                         cons ('source,
                                               value (cadr (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk))))),
                                  tag ('pc,
                                       cons (subr, length (code (cinfo))
                                                   +  3))),
                          ctrl-stk),
                    push (value (cadr (call-actuals (stmt)),
                                 bindings (top (ctrl-stk))),
                          map-down-values (mg-alist (mg-state),
                                           bindings (top (ctrl-stk)),
                                           temp-stk)),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
     =  p-state (tag ('pc,
```

```
        '(mg-simple-variable-assignment . 2)),
    push (p-frame (list (cons ('dest,
                               value (car (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                          cons ('source,
                               value (cadr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk))))),
                      tag ('pc,
                          cons (subr, length (code (cinfo))
                                      + 3))),
          ctrl-stk),
    push (rget (untag (value (cadr (call-actuals (stmt)),
                               bindings (top (ctrl-stk)))),
                      map-down-values (mg-alist (mg-state),
                                       bindings (top (ctrl-stk)),
                                       temp-stk)),
          map-down-values (mg-alist (mg-state),
                           bindings (top (ctrl-stk)),
                           temp-stk)),
    translate-proc-list (proc-list),
    list (list ('c-c,
                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
    MG-MAX-CTRL-STK-SIZE,
    MG-MAX-TEMP-STK-SIZE,
    MG-WORD-SIZE,
    'run))

;; (push-local dest)
```

THEOREM: mg-simple-variable-assignment-step-6

```
((¬ resources-inadequatep (stmt,
                           proc-list,
                           list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'predefined-proc-call-mg)
 ∧  (call-name (stmt) = 'mg-simple-variable-assignment)
 ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧  ok-mg-def-plistp (proc-list)
 ∧  ok-mg-statep (mg-state, r-cond-list)
 ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧  user-defined-procp (subr, proc-list)
 ∧  all-cars-unique (mg-alist (mg-state))
```

```
∧  signatures-match (mg-alist (mg-state), name-alist)
∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
∧  normal (mg-state))
→  (p-step (p-state (tag ('pc,
                          '(mg-simple-variable-assignment . 2)),
                    push (p-frame (list (cons ('dest,
                                               value (car (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk)))),
                                         cons ('source,
                                               value (cadr (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk))))),
                                   tag ('pc,
                                        cons (subr, length (code (cinfo))
                                                    + 3))),
                          ctrl-stk),
                    push (rget (untag (value (cadr (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk)))),
                                       map-down-values (mg-alist (mg-state),
                                                        bindings (top (ctrl-stk)),
                                                        temp-stk)),
                               map-down-values (mg-alist (mg-state),
                                                bindings (top (ctrl-stk)),
                                                temp-stk)),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
   =  p-state (tag ('pc,
                    '(mg-simple-variable-assignment . 3)),
               push (p-frame (list (cons ('dest,
                                          value (car (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk)))),
                                    cons ('source,
                                          value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk))))),
                              tag ('pc,
                                   cons (subr, length (code (cinfo))
                                               + 3))),
                     ctrl-stk),
```

```
                push (value (car (call-actuals (stmt)),
                                 bindings (top (ctrl-stk))),
                     push (rget (untag (value (cadr (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk)))),
                                       map-down-values (mg-alist (mg-state),
                                                        bindings (top (ctrl-stk)),
                                                        temp-stk)),
                                 map-down-values (mg-alist (mg-state),
                                                  bindings (top (ctrl-stk)),
                                                  temp-stk))),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))

;; (deposit-temp-stk)


THEOREM: mg-simple-variable-assignment-step-7
 ((¬ resources-inadequatep (stmt,
                            proc-list,
                            list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
  ∧  (car (stmt) = 'predefined-proc-call-mg)
  ∧  (call-name (stmt) = 'mg-simple-variable-assignment)
  ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
  ∧  ok-mg-def-plistp (proc-list)
  ∧  ok-mg-statep (mg-state, r-cond-list)
  ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
      =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                 code2))
  ∧  user-defined-procp (subr, proc-list)
  ∧  all-cars-unique (mg-alist (mg-state))
  ∧  signatures-match (mg-alist (mg-state), name-alist)
  ∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                 bindings (top (ctrl-stk)),
                                 temp-stk)
  ∧  normal (mg-state))
 →  (p-step (p-state (tag ('pc,
                           '(mg-simple-variable-assignment . 3)),
                     push (p-frame (list (cons ('dest,
                                               value (car (call-actuals (stmt)),
```

```
                                                bindings (top (ctrl-stk)))),
                                        cons ('source,
                                              value (cadr (call-actuals (stmt)),
                                                     bindings (top (ctrl-stk))))),
                                    tag ('pc,
                                         cons (subr, length (code (cinfo))
                                                     +  3))),
                                ctrl-stk),
                            push (value (car (call-actuals (stmt)),
                                             bindings (top (ctrl-stk))),
                                  push (rget (untag (value (cadr (call-actuals (stmt)),
                                                           bindings (top (ctrl-stk)))),
                                                    map-down-values (mg-alist (mg-state),
                                                                     bindings (top (ctrl-stk)),
                                                                     temp-stk)),
                                             map-down-values (mg-alist (mg-state),
                                                              bindings (top (ctrl-stk)),
                                                              temp-stk))),
                            translate-proc-list (proc-list),
                            list (list ('c-c,
                                        mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                            MG-MAX-CTRL-STK-SIZE,
                            MG-MAX-TEMP-STK-SIZE,
                            MG-WORD-SIZE,
                            'run))
=   p-state (tag ('pc,
                  '(mg-simple-variable-assignment . 4)),
             push (p-frame (list (cons ('dest,
                                        value (car (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                       cons ('source,
                                             value (cadr (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk))))),
                                 tag ('pc,
                                      cons (subr, length (code (cinfo))
                                                  +  3))),
                           ctrl-stk),
             rput (rget (untag (value (cadr (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                               map-down-values (mg-alist (mg-state),
                                                bindings (top (ctrl-stk)),
                                                temp-stk)),
                        untag (value (car (call-actuals (stmt)),
                                          bindings (top (ctrl-stk)))),
```

```
                map-down-values (mg-alist (mg-state),
                                           bindings (top (ctrl-stk)),
                                           temp-stk)),
           translate-proc-list (proc-list),
           list (list ('c-c,
                      mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
           MG-MAX-CTRL-STK-SIZE,
           MG-MAX-TEMP-STK-SIZE,
           MG-WORD-SIZE,
           'run))

;; (ret)
```

THEOREM: mg-simple-variable-assignment-step-8

$$
\begin{array}{ll}
 & ((n \not\simeq \texttt{0}) \\
\wedge & (\neg\ \text{resources-inadequatep}\ (stmt, \\
 & \qquad\qquad proc\text{-}list, \\
 & \qquad\qquad \text{list}\ (\text{length}\ (temp\text{-}stk), \\
 & \qquad\qquad\quad \text{p-ctrl-stk-size}\ (ctrl\text{-}stk)))) \\
\wedge & (\text{car}\ (stmt) = \texttt{'predefined-proc-call-mg}) \\
\wedge & (\text{call-name}\ (stmt) = \texttt{'mg-simple-variable-assignment}) \\
\wedge & \text{ok-mg-statement}\ (stmt,\ r\text{-}cond\text{-}list,\ name\text{-}alist,\ proc\text{-}list) \\
\wedge & \text{ok-mg-def-plistp}\ (proc\text{-}list) \\
\wedge & \text{ok-mg-statep}\ (mg\text{-}state,\ r\text{-}cond\text{-}list) \\
\wedge & (\text{code}\ (\text{translate-def-body}\ (\text{assoc}\ (subr,\ proc\text{-}list),\ proc\text{-}list)) \\
 & \quad = \quad \text{append}\ (\text{code}\ (\text{translate}\ (cinfo,\ t\text{-}cond\text{-}list,\ stmt,\ proc\text{-}list)), \\
 & \qquad\quad code2)) \\
\wedge & \text{user-defined-procp}\ (subr,\ proc\text{-}list) \\
\wedge & \text{listp}\ (ctrl\text{-}stk) \\
\wedge & \text{all-cars-unique}\ (\text{mg-alist}\ (mg\text{-}state)) \\
\wedge & \text{signatures-match}\ (\text{mg-alist}\ (mg\text{-}state),\ name\text{-}alist) \\
\wedge & \text{mg-vars-list-ok-in-p-state}\ (\text{mg-alist}\ (mg\text{-}state), \\
 & \qquad\qquad bindings\ (\text{top}\ (ctrl\text{-}stk)), \\
 & \qquad\qquad temp\text{-}stk) \\
\wedge & \text{no-p-aliasing}\ (\text{bindings}\ (\text{top}\ (ctrl\text{-}stk)),\ \text{mg-alist}\ (mg\text{-}state)) \\
\wedge & \text{normal}\ (mg\text{-}state)) \\
\rightarrow & (\text{p-step}\ (\text{p-state}\ (\text{tag}\ (\texttt{'pc}, \\
 & \qquad\qquad\qquad \texttt{'(mg-simple-variable-assignment . 4)}), \\
 & \qquad\qquad \text{push}\ (\text{p-frame}\ (\text{list}\ (\text{cons}\ (\texttt{'dest}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \text{value}\ (\text{car}\ (\text{call-actuals}\ (stmt)), \\
 & \qquad\qquad\qquad\qquad\qquad\qquad\quad \text{bindings}\ (\text{top}\ (ctrl\text{-}stk)))), \\
 & \qquad\qquad\qquad\qquad\quad \text{cons}\ (\texttt{'source}, \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \text{value}\ (\text{cadr}\ (\text{call-actuals}\ (stmt)),
\end{array}
$$

```
                                                    bindings (top (ctrl-stk)))))),
                                        tag ('pc,
                                             cons (subr, length (code (cinfo))
                                                         +    3))),
                                   ctrl-stk),
                              rput (rget (untag (value (cadr (call-actuals (stmt)),
                                                             bindings (top (ctrl-stk)))),
                                                 map-down-values (mg-alist (mg-state),
                                                                  bindings (top (ctrl-stk)),
                                                                  temp-stk)),
                                         untag (value (car (call-actuals (stmt)),
                                                           bindings (top (ctrl-stk)))),
                                         map-down-values (mg-alist (mg-state),
                                                          bindings (top (ctrl-stk)),
                                                          temp-stk)),
                              translate-proc-list (proc-list),
                              list (list ('c-c,
                                          mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                              MG-MAX-CTRL-STK-SIZE,
                              MG-MAX-TEMP-STK-SIZE,
                              MG-WORD-SIZE,
                              'run))
=    p-state (tag ('pc,
                   cons (subr,
                         if normal (mg-meaning-r (stmt,
                                                  proc-list,
                                                  mg-state,
                                                  n,
                                                  list (length (temp-stk),
                                                        p-ctrl-stk-size (ctrl-stk))))
                         then length (code (translate (cinfo,
                                                       t-cond-list,
                                                       stmt,
                                                       proc-list)))
                         else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                         proc-list,
                                                                         mg-state,
                                                                         n,
                                                                         list (length (temp-stk),
                                                                               p-ctrl-stk-size (ctrl-stk)))),
                                                       label-alist (translate (cinfo,
                                                                               t-cond-list,
                                                                               stmt,
                                                                               proc-list))),
```

append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-variable-assignment-exact-time-lemma
((*n* ≉ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-variable-assignment)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))

```
∧  user-defined-procp (subr, proc-list)
∧  plistp (temp-stk)
∧  listp (ctrl-stk)
∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
∧  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
∧  signatures-match (mg-alist (mg-state), name-alist)
∧  normal (mg-state)
∧  all-cars-unique (mg-alist (mg-state))
∧  (¬ resource-errorp (mg-meaning-r (stmt,
                                     proc-list,
                                     mg-state,
                                     n,
                                     list (length (temp-stk),
                                           p-ctrl-stk-size (ctrl-stk))))))
→  (p (map-down (mg-state,
                proc-list,
                ctrl-stk,
                temp-stk,
                tag ('pc, cons (subr, length (code (cinfo)))),
                t-cond-list),
      clock (stmt, proc-list, mg-state, n))
   =  p-state (tag ('pc,
                    cons (subr,
                          if normal (mg-meaning-r (stmt,
                                                   proc-list,
                                                   mg-state,
                                                   n,
                                                   list (length (temp-stk),
                                                         p-ctrl-stk-size (ctrl-stk))))
                          then length (code (translate (cinfo,
                                                        t-cond-list,
                                                        stmt,
                                                        proc-list)))
                          else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                          proc-list,
                                                                          mg-state,
                                                                          n,
                                                                          list (length (temp-stk),
                                                                                p-ctrl-stk-size (ctrl-stk)))),
                                                        label-alist (translate (cinfo,
                                                                                t-cond-list,
                                                                                stmt,
```

proc-list))),
append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                              ;;
;;              EXACT-TIME LEMMA MG-SIMPLE-CONSTANT-ASSIGNMENT                  ;;
;;                                                                              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: mg-simple-constant-assignment-args-simple-identifierps
(ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-constant-assignment)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ signatures-match (mg-alist (*mg-state*), *name-alist*))
→ simple-identifierp (car (call-actuals (*stmt*)), mg-alist (*mg-state*))

THEOREM: mg-simple-constant-assignment-steps-1-2

((¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*), p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-assignment`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ normal (*mg-state*))
→ (p-step (p-step (map-down (*mg-state*,
proc-list,
ctrl-stk,
temp-stk,
tag (`'pc`, cons (*subr*, length (code (*cinfo*)))),
t-cond-list)))
= p-state (tag (`'pc`, cons (*subr*, length (code (*cinfo*)) + `2`)),
ctrl-stk,
push (mg-to-p-simple-literal (cadr (call-actuals (*stmt*))),
push (value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

```
;; (call mg-simple-constant-assignment)
```

THEOREM: mg-simple-constant-assignment-step-3
((¬ resources-inadequatep (*stmt*,
proc-list,

```
                                  list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
∧   (car (stmt) = 'predefined-proc-call-mg)
∧   (call-name (stmt) = 'mg-simple-constant-assignment)
∧   ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
∧   ok-mg-def-plistp (proc-list)
∧   ok-mg-statep (mg-state, r-cond-list)
∧   (code (translate-def-body (assoc (subr, proc-list), proc-list))
      =   append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                       code2))
∧   user-defined-procp (subr, proc-list)
∧   mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                  bindings (top (ctrl-stk)),
                                  temp-stk)
∧   normal (mg-state))
→   (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 2)),
                      ctrl-stk,
                      push (mg-to-p-simple-literal (cadr (call-actuals (stmt))),
                            push (value (car (call-actuals (stmt)),
                                        bindings (top (ctrl-stk))),
                                 map-down-values (mg-alist (mg-state),
                                                  bindings (top (ctrl-stk)),
                                                  temp-stk))),
                      translate-proc-list (proc-list),
                      list (list ('c-c,
                                  mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                      MG-MAX-CTRL-STK-SIZE,
                      MG-MAX-TEMP-STK-SIZE,
                      MG-WORD-SIZE,
                      'run))
      =   p-state (tag ('pc,
                        '(mg-simple-constant-assignment . 0)),
                   push (p-frame (list (cons ('dest,
                                              value (car (call-actuals (stmt)),
                                                     bindings (top (ctrl-stk))))),
                                       cons ('source,
                                             mg-to-p-simple-literal (cadr (call-actuals (stmt))))),
                                 tag ('pc,
                                      cons (subr, length (code (cinfo))
                                                  +   3))),
                         ctrl-stk),
                   map-down-values (mg-alist (mg-state),
                                    bindings (top (ctrl-stk)),
                                    temp-stk),
                   translate-proc-list (proc-list),
```

```
                    list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))

;; (push-local source)
```

THEOREM: mg-simple-constant-assignment-steps-4-5

```
((¬ resources-inadequatep (stmt,
                           proc-list,
                           list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'predefined-proc-call-mg)
 ∧  (call-name (stmt) = 'mg-simple-constant-assignment)
 ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧  ok-mg-def-plistp (proc-list)
 ∧  ok-mg-statep (mg-state, r-cond-list)
 ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧  user-defined-procp (subr, proc-list)
 ∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 ∧  normal (mg-state))
→  (p-step (p-step (p-state (tag ('pc,
                                  '(mg-simple-constant-assignment
                                    . 0)),
                             push (p-frame (list (cons ('dest,
                                                        value (car (call-actuals (stmt)),
                                                               bindings (top (ctrl-stk))))),
                                                 cons ('source,
                                                       mg-to-p-simple-literal (cadr (call-actuals (stmt))))),
                                           tag ('pc,
                                                cons (subr,
                                                      length (code (cinfo))
                                                      +  3))),
                                   ctrl-stk),
                             map-down-values (mg-alist (mg-state),
                                              bindings (top (ctrl-stk)),
                                              temp-stk),
                             translate-proc-list (proc-list),
```

```
                    list (list ('c-c,
                               mg-cond-to-p-nat (cc (mg-state),
                                                    t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run)))
    =  p-state (tag ('pc,
                     '(mg-simple-constant-assignment . 2)),
                push (p-frame (list (cons ('dest,
                                           value (car (call-actuals (stmt)),
                                                  bindings (top (ctrl-stk)))),
                                          cons ('source,
                                                mg-to-p-simple-literal (cadr (call-actuals (stmt))))),
                                    tag ('pc,
                                         cons (subr, length (code (cinfo))
                                                     +  3))),
                              ctrl-stk),
                push (value (car (call-actuals (stmt)),
                             bindings (top (ctrl-stk))),
                      push (mg-to-p-simple-literal (cadr (call-actuals (stmt))),
                            map-down-values (mg-alist (mg-state),
                                             bindings (top (ctrl-stk)),
                                             temp-stk))),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))

;; (deposit-temp-stk)
```

THEOREM: mg-simple-constant-assignment-step-6

```
 ((¬ resources-inadequatep (stmt,
                            proc-list,
                            list (length (temp-stk), p-ctrl-stk-size (ctrl-stk))))
  ∧  (car (stmt) = 'predefined-proc-call-mg)
  ∧  (call-name (stmt) = 'mg-simple-constant-assignment)
  ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
  ∧  ok-mg-def-plistp (proc-list)
  ∧  ok-mg-statep (mg-state, r-cond-list)
```

```
∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                     code2))
∧  user-defined-procp (subr, proc-list)
∧  all-cars-unique (mg-alist (mg-state))
∧  signatures-match (mg-alist (mg-state), name-alist)
∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
∧  normal (mg-state))
→  (p-step (p-state (tag ('pc,
                          '(mg-simple-constant-assignment . 2)),
                    push (p-frame (list (cons ('dest,
                                               value (car (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk))))),
                                        cons ('source,
                                              mg-to-p-simple-literal (cadr (call-actuals (stmt))))),
                                  tag ('pc,
                                       cons (subr, length (code (cinfo))
                                                   +  3))),
                          ctrl-stk),
                    push (value (car (call-actuals (stmt)),
                                 bindings (top (ctrl-stk))),
                          push (mg-to-p-simple-literal (cadr (call-actuals (stmt))),
                                map-down-values (mg-alist (mg-state),
                                                 bindings (top (ctrl-stk)),
                                                 temp-stk))),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
     =  p-state (tag ('pc,
                      '(mg-simple-constant-assignment . 3)),
                 push (p-frame (list (cons ('dest,
                                            value (car (call-actuals (stmt)),
                                                   bindings (top (ctrl-stk))))),
                                     cons ('source,
                                           mg-to-p-simple-literal (cadr (call-actuals (stmt))))),
                               tag ('pc,
                                    cons (subr, length (code (cinfo))
                                                +  3))),
```

ctrl-stk),
rput (mg-to-p-simple-literal (cadr (call-actuals (*stmt*))),
untag (value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

```
;; (ret)
```

THEOREM: mg-simple-constant-assignment-step-7
((*n* ≉ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-constant-assignment)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ listp (*ctrl-stk*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc,
'(mg-simple-constant-assignment . 3)),
push (p-frame (list (cons ('dest,

```
                                            value (car (call-actuals (stmt)),
                                                   bindings (top (ctrl-stk)))),
                                      cons ('source,
                                            mg-to-p-simple-literal (cadr (call-actuals (stmt))))),
                                 tag ('pc,
                                      cons (subr, length (code (cinfo))
                                                  +   3))),
                        ctrl-stk),
                 rput (mg-to-p-simple-literal (cadr (call-actuals (stmt))),
                       untag (value (car (call-actuals (stmt)),
                                      bindings (top (ctrl-stk)))),
                       map-down-values (mg-alist (mg-state),
                                           bindings (top (ctrl-stk)),
                                           temp-stk)),
                 translate-proc-list (proc-list),
                 list (list ('c-c,
                          mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                 MG-MAX-CTRL-STK-SIZE,
                 MG-MAX-TEMP-STK-SIZE,
                 MG-WORD-SIZE,
                 'run))
=    p-state (tag ('pc,
                  cons (subr,
                        if normal (mg-meaning-r (stmt,
                                                 proc-list,
                                                 mg-state,
                                                 n,
                                                 list (length (temp-stk),
                                                       p-ctrl-stk-size (ctrl-stk))))
                        then length (code (translate (cinfo,
                                                      t-cond-list,
                                                      stmt,
                                                      proc-list)))
                        else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                         proc-list,
                                                                         mg-state,
                                                                         n,
                                                                         list (length (temp-stk),
                                                                               p-ctrl-stk-size (ctrl-stk)))),
                                                      label-alist (translate (cinfo,
                                                                              t-cond-list,
                                                                              stmt,
                                                                              proc-list))),
                                          append (code (translate (cinfo,
```

t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-constant-assignment-exact-time-lemma
((*n* ≠ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-assignment`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-translation-parameters (*cinfo*, *t-cond-list*, *stmt*, *proc-list*, *code2*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ cond-subsetp (*r-cond-list*, *t-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)

$\wedge$ plistp (*temp-stk*)
$\wedge$ listp (*ctrl-stk*)
$\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
$\wedge$ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*)
$\wedge$ normal (*mg-state*)
$\wedge$ all-cars-unique (mg-alist (*mg-state*))
$\wedge$ ($\neg$ resource-errorp (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))))
$\rightarrow$ (p (map-down (*mg-state*,
proc-list,
ctrl-stk,
temp-stk,
tag ('pc, cons (*subr*, length (code (*cinfo*)))),
t-cond-list),
clock (*stmt*, *proc-list*, *mg-state*, *n*))
= p-state (tag ('pc,
cons (*subr*,
if normal (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
then length (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)))
else find-label (fetch-label (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
label-alist (translate (*cinfo*,
t-cond-list,
stmt,
proc-list))),

```
                                                append (code (translate (cinfo,
                                                                         t-cond-list,
                                                                         stmt,
                                                                         proc-list)),
                                                             code2)) endif)),
                                ctrl-stk,
                                map-down-values (mg-alist (mg-meaning-r (stmt,
                                                                        proc-list,
                                                                        mg-state,
                                                                        n,
                                                                        list (length (temp-stk),
                                                                              p-ctrl-stk-size (ctrl-stk)))),
                                                           bindings (top (ctrl-stk)),
                                                           temp-stk),
                                translate-proc-list (proc-list),
                                list (list ('c-c,
                                            mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                                               proc-list,
                                                                               mg-state,
                                                                               n,
                                                                               list (length (temp-stk),
                                                                                     p-ctrl-stk-size (ctrl-stk)))),
                                                                 t-cond-list))),
                                MG-MAX-CTRL-STK-SIZE,
                                MG-MAX-TEMP-STK-SIZE,
                                MG-WORD-SIZE,
                                'run))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                                    ;;
;;                  EXACT-TIME LEMMA MG-SIMPLE-CONSTANT-EQ                            ;;
;;                                                                                    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: mg-simple-variable-eq-args-simple-identifierps
(ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
 ∧ (car (*stmt*) = 'predefined-proc-call-mg)
 ∧ (call-name (*stmt*) = 'mg-simple-variable-eq)
 ∧ ok-mg-statep (*mg-state*, *r-cond-list*)
 ∧ signatures-match (mg-alist (*mg-state*), *name-alist*))
→ (boolean-identifierp (car (call-actuals (*stmt*)), mg-alist (*mg-state*))
 ∧ simple-identifierp (cadr (call-actuals (*stmt*)),
 mg-alist (*mg-state*))

$\wedge$ simple-identifierp (caddr (call-actuals (*stmt*)),
mg-alist (*mg-state*)))

THEOREM: mg-simple-variable-eq-args-definedp
(ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ (car (*stmt*) = `'predefined-proc-call-mg`)
$\wedge$ (call-name (*stmt*) = `'mg-simple-variable-eq`)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*))
$\rightarrow$ (definedp (car (call-actuals (*stmt*)), mg-alist (*mg-state*))
$\wedge$ definedp (cadr (call-actuals (*stmt*)), mg-alist (*mg-state*))
$\wedge$ definedp (caddr (call-actuals (*stmt*)), mg-alist (*mg-state*)))

THEOREM: mg-simple-variable-eq-steps-1-3
((*n* $\not\simeq$ `0`)
$\wedge$ ($\neg$ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
$\wedge$ (car (*stmt*) = `'predefined-proc-call-mg`)
$\wedge$ (call-name (*stmt*) = `'mg-simple-variable-eq`)
$\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ ok-mg-def-plistp (*proc-list*)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
$\wedge$ user-defined-procp (*subr*, *proc-list*)
$\wedge$ listp (*ctrl-stk*)
$\wedge$ all-cars-unique (mg-alist (*mg-state*))
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*)
$\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
$\wedge$ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
$\wedge$ normal (*mg-state*))
$\rightarrow$ (p-step (p-step (p-step (map-down (*mg-state*,
proc-list,
ctrl-stk,
temp-stk,
tag (`'pc`,
cons (*subr*, length (code (*cinfo*)))),
t-cond-list))))
= p-state (tag (`'pc`, cons (*subr*, length (code (*cinfo*)) + `3`)),

ctrl-stk,
push (value (caddr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
push (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
push (value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)))),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-variable-eq-step-4
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-variable-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc, cons (*subr*, length (code (*cinfo*)) + 3)),
ctrl-stk,
push (value (caddr (call-actuals (*stmt*)),

```
                                    bindings (top (ctrl-stk))),
                                push (value (cadr (call-actuals (stmt)),
                                            bindings (top (ctrl-stk))),
                                      push (value (car (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk))),
                                            map-down-values (mg-alist (mg-state),
                                                                       bindings (top (ctrl-stk)),
                                                                       temp-stk)))),
                           translate-proc-list (proc-list),
                           list (list ('c-c,
                                       mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                           MG-MAX-CTRL-STK-SIZE,
                           MG-MAX-TEMP-STK-SIZE,
                           MG-WORD-SIZE,
                           'run))
    =  p-state (tag ('pc, '(mg-simple-variable-eq . 0)),
                push (p-frame (list (cons ('ans,
                                           value (car (call-actuals (stmt)),
                                                  bindings (top (ctrl-stk)))),
                                          cons ('x,
                                                value (cadr (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk)))),
                                          cons ('y,
                                                value (caddr (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk))))),
                                    tag ('pc,
                                         cons (subr, length (code (cinfo))
                                                     + 4))),
                      ctrl-stk),
                map-down-values (mg-alist (mg-state),
                                           bindings (top (ctrl-stk)),
                                           temp-stk),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))

THEOREM: mg-simple-variable-eq-step-5
((n ≠ 0)
 ∧  (¬ resources-inadequatep (stmt,
                              proc-list,
```

```
                                    list (length (temp-stk),
                                          p-ctrl-stk-size (ctrl-stk))))
  ∧   (car (stmt) = 'predefined-proc-call-mg)
  ∧   (call-name (stmt) = 'mg-simple-variable-eq)
  ∧   ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
  ∧   ok-mg-def-plistp (proc-list)
  ∧   ok-mg-statep (mg-state, r-cond-list)
  ∧   (code (translate-def-body (assoc (subr, proc-list), proc-list))
      =   append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                       code2))
  ∧   user-defined-procp (subr, proc-list)
  ∧   listp (ctrl-stk)
  ∧   all-cars-unique (mg-alist (mg-state))
  ∧   signatures-match (mg-alist (mg-state), name-alist)
  ∧   mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                  bindings (top (ctrl-stk)),
                                  temp-stk)
  ∧   no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
  ∧   normal (mg-state))
  →   (p-step (p-state (tag ('pc, '(mg-simple-variable-eq . 0)),
                       push (p-frame (list (cons 'ans,
                                                 value (car (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk)))),
                                           cons ('x,
                                                 value (cadr (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk)))),
                                           cons ('y,
                                                 value (caddr (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk))))),
                                     tag ('pc,
                                          cons (subr, length (code (cinfo))
                                                      +   4))),
                            ctrl-stk),
                       map-down-values (mg-alist (mg-state),
                                        bindings (top (ctrl-stk)),
                                        temp-stk),
                       translate-proc-list (proc-list),
                       list (list ('c-c,
                                   mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                       MG-MAX-CTRL-STK-SIZE,
                       MG-MAX-TEMP-STK-SIZE,
                       MG-WORD-SIZE,
                       'run))
      =   p-state (tag ('pc, '(mg-simple-variable-eq . 1)),
```

```
push (p-frame (list (cons ('ans,
                           value (car (call-actuals (stmt)),
                                  bindings (top (ctrl-stk)))),
                    cons ('x,
                           value (cadr (call-actuals (stmt)),
                                  bindings (top (ctrl-stk)))),
                    cons ('y,
                           value (caddr (call-actuals (stmt)),
                                  bindings (top (ctrl-stk))))),
              tag ('pc,
                   cons (subr, length (code (cinfo))
                               +  4))),
      ctrl-stk),
push (value (cadr (call-actuals (stmt)),
            bindings (top (ctrl-stk))),
      map-down-values (mg-alist (mg-state),
                       bindings (top (ctrl-stk)),
                       temp-stk)),
translate-proc-list (proc-list),
list (list ('c-c,
        mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))
```

THEOREM: mg-simple-variable-eq-step-6

$$
\begin{aligned}
&((n \not\simeq \texttt{0})\\
&\wedge\ (\neg\ \text{resources-inadequatep}\ (stmt,\\
&\qquad\qquad proc\text{-}list,\\
&\qquad\qquad \text{list}\ (\text{length}\ (temp\text{-}stk),\\
&\qquad\qquad\quad \text{p-ctrl-stk-size}\ (ctrl\text{-}stk))))\\
&\wedge\ (\text{car}\ (stmt) = \texttt{'predefined-proc-call-mg})\\
&\wedge\ (\text{call-name}\ (stmt) = \texttt{'mg-simple-variable-eq})\\
&\wedge\ \text{ok-mg-statement}\ (stmt,\ r\text{-}cond\text{-}list,\ name\text{-}alist,\ proc\text{-}list)\\
&\wedge\ \text{ok-mg-def-plistp}\ (proc\text{-}list)\\
&\wedge\ \text{ok-mg-statep}\ (mg\text{-}state,\ r\text{-}cond\text{-}list)\\
&\wedge\ (\text{code}\ (\text{translate-def-body}\ (\text{assoc}\ (subr,\ proc\text{-}list),\ proc\text{-}list))\\
&\quad =\ \ \text{append}\ (\text{code}\ (\text{translate}\ (cinfo,\ t\text{-}cond\text{-}list,\ stmt,\ proc\text{-}list)),\\
&\qquad\qquad code2))\\
&\wedge\ \text{user-defined-procp}\ (subr,\ proc\text{-}list)\\
&\wedge\ \text{listp}\ (ctrl\text{-}stk)\\
&\wedge\ \text{all-cars-unique}\ (\text{mg-alist}\ (mg\text{-}state))\\
&\wedge\ \text{signatures-match}\ (\text{mg-alist}\ (mg\text{-}state),\ name\text{-}alist)
\end{aligned}
$$

```
∧   mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
∧   no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
∧   normal (mg-state))
→   (p-step (p-state (tag ('pc, '(mg-simple-variable-eq . 1)),
                      push (p-frame (list (cons ('ans,
                                                value (car (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk)))),
                                                cons ('x,
                                                      value (cadr (call-actuals (stmt)),
                                                             bindings (top (ctrl-stk)))),
                                                cons ('y,
                                                      value (caddr (call-actuals (stmt)),
                                                             bindings (top (ctrl-stk))))),
                                          tag ('pc,
                                               cons (subr, length (code (cinfo))
                                                           +  4))),
                                    ctrl-stk),
                      push (value (cadr (call-actuals (stmt)),
                                  bindings (top (ctrl-stk))),
                            map-down-values (mg-alist (mg-state),
                                             bindings (top (ctrl-stk)),
                                             temp-stk)),
                      translate-proc-list (proc-list),
                      list (list ('c-c,
                                  mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                      MG-MAX-CTRL-STK-SIZE,
                      MG-MAX-TEMP-STK-SIZE,
                      MG-WORD-SIZE,
                      'run))
    =   p-state (tag ('pc, '(mg-simple-variable-eq . 2)),
                 push (p-frame (list (cons ('ans,
                                            value (car (call-actuals (stmt)),
                                                   bindings (top (ctrl-stk)))),
                                            cons ('x,
                                                  value (cadr (call-actuals (stmt)),
                                                         bindings (top (ctrl-stk)))),
                                            cons ('y,
                                                  value (caddr (call-actuals (stmt)),
                                                         bindings (top (ctrl-stk))))),
                                      tag ('pc,
                                           cons (subr, length (code (cinfo))
                                                       +  4))),
```

ctrl-stk),
push (rget (untag (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
push (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-variable-eq-step-7
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-variable-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc, '(mg-simple-variable-eq . 2)),
push (p-frame (list (cons ('ans,

```
                                        value (car (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                    cons ('x,
                                        value (cadr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                    cons ('y,
                                        value (caddr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk))))),
                                tag ('pc,
                                    cons (subr, length (code (cinfo))
                                                + 4))),
                            ctrl-stk),
                        push (rget (untag (value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk)))),
                                   push (value (cadr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))),
                                         map-down-values (mg-alist (mg-state),
                                                          bindings (top (ctrl-stk)),
                                                          temp-stk))),
                                   map-down-values (mg-alist (mg-state),
                                                    bindings (top (ctrl-stk)),
                                                    temp-stk)),
                                   translate-proc-list (proc-list),
                                   list (list ('c-c,
                                               mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                                   MG-MAX-CTRL-STK-SIZE,
                                   MG-MAX-TEMP-STK-SIZE,
                                   MG-WORD-SIZE,
                                   'run))
=    p-state (tag ('pc, '(mg-simple-variable-eq . 3)),
                  push (p-frame (list (cons ('ans,
                                             value (car (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk)))),
                                      cons ('x,
                                          value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk)))),
                                      cons ('y,
                                          value (caddr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk))))),
                                  tag ('pc,
                                      cons (subr, length (code (cinfo))
                                                  + 4))),
                          ctrl-stk),
                  push (value (caddr (call-actuals (stmt)),
```

bindings (top (*ctrl-stk*))),
push (rget (untag (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
push (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
translate-proc-list (*proc-list*),
list (list ('`c-c`,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'`run`))

THEOREM: mg-simple-variable-eq-step-8
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = '`predefined-proc-call-mg`)
∧ (call-name (*stmt*) = '`mg-simple-variable-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('`pc`, '`(mg-simple-variable-eq . 3)`),
push (p-frame (list (cons ('`ans`,

```
                                        value (car (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                    cons ('x,
                                        value (cadr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                    cons ('y,
                                        value (caddr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk))))),
                                tag ('pc,
                                    cons (subr, length (code (cinfo))
                                                + 4))),
                            ctrl-stk),
                        push (value (caddr (call-actuals (stmt)),
                                    bindings (top (ctrl-stk))),
                            push (rget (untag (value (cadr (call-actuals (stmt)),
                                                     bindings (top (ctrl-stk)))),
                                        push (value (cadr (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk))),
                                            map-down-values (mg-alist (mg-state),
                                                             bindings (top (ctrl-stk)),
                                                             temp-stk))),
                                    map-down-values (mg-alist (mg-state),
                                                     bindings (top (ctrl-stk)),
                                                     temp-stk))),
                        translate-proc-list (proc-list),
                        list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                        MG-MAX-CTRL-STK-SIZE,
                        MG-MAX-TEMP-STK-SIZE,
                        MG-WORD-SIZE,
                        'run))
=   p-state (tag ('pc, '(mg-simple-variable-eq . 4)),
            push (p-frame (list (cons ('ans,
                                       value (car (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                  cons ('x,
                                       value (cadr (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                  cons ('y,
                                       value (caddr (call-actuals (stmt)),
                                              bindings (top (ctrl-stk))))),
                              tag ('pc,
                                  cons (subr, length (code (cinfo))
                                              + 4))),
```

ctrl-stk),
push (rget (untag (value (caddr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
push (rget (untag (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-variable-eq-step-9
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-variable-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))

```
  ∧   normal (mg-state))
  →   (p-step (p-state (tag ('pc, '(mg-simple-variable-eq . 4)),
                       push (p-frame (list (cons ('ans,
                                                 value (car (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk))))),
                                           cons ('x,
                                                 value (cadr (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk))))),
                                           cons ('y,
                                                 value (caddr (call-actuals (stmt)),
                                                        bindings (top (ctrl-stk)))))),
                                     tag ('pc,
                                          cons (subr, length (code (cinfo))
                                                      +    4))),
                                ctrl-stk),
                       push (rget (untag (value (caddr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                         map-down-values (mg-alist (mg-state),
                                                          bindings (top (ctrl-stk)),
                                                          temp-stk)),
                                  push (rget (untag (value (cadr (call-actuals (stmt)),
                                                           bindings (top (ctrl-stk)))),
                                                    map-down-values (mg-alist (mg-state),
                                                                     bindings (top (ctrl-stk)),
                                                                     temp-stk)),
                                             map-down-values (mg-alist (mg-state),
                                                              bindings (top (ctrl-stk)),
                                                              temp-stk))),
                       translate-proc-list (proc-list),
                       list (list ('c-c,
                                   mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                       MG-MAX-CTRL-STK-SIZE,
                       MG-MAX-TEMP-STK-SIZE,
                       MG-WORD-SIZE,
                       'run))
  =   p-state (tag ('pc, '(mg-simple-variable-eq . 5)),
               push (p-frame (list (cons ('ans,
                                          value (car (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk))))),
                                   cons ('x,
                                         value (cadr (call-actuals (stmt)),
                                                bindings (top (ctrl-stk))))),
                                   cons ('y,
                                         value (caddr (call-actuals (stmt)),
```

```
                                                    bindings (top (ctrl-stk))))),
                                    tag ('pc,
                                        cons (subr, length (code (cinfo))
                                                    +   4))),
                            ctrl-stk),
                    push (tag ('bool,
                            if untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                           mg-alist (mg-state)))))
                              =   untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (stmt)),
                                                                              mg-alist (mg-state)))))
                            then 't
                            else 'f endif),
                        map-down-values (mg-alist (mg-state),
                                         bindings (top (ctrl-stk)),
                                         temp-stk)),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
```

THEOREM: mg-simple-variable-eq-step-10

```
((n ≠ 0)
 ∧   (¬ resources-inadequatep (stmt,
                               proc-list,
                               list (length (temp-stk),
                                     p-ctrl-stk-size (ctrl-stk))))
 ∧   (car (stmt) = 'predefined-proc-call-mg)
 ∧   (call-name (stmt) = 'mg-simple-variable-eq)
 ∧   ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧   ok-mg-def-plistp (proc-list)
 ∧   ok-mg-statep (mg-state, r-cond-list)
 ∧   (code (translate-def-body (assoc (subr, proc-list), proc-list))
       =   append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                   code2))
 ∧   user-defined-procp (subr, proc-list)
 ∧   listp (ctrl-stk)
 ∧   all-cars-unique (mg-alist (mg-state))
 ∧   signatures-match (mg-alist (mg-state), name-alist)
 ∧   mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                 bindings (top (ctrl-stk)),
                                 temp-stk)
```

```
  ∧  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
  ∧  normal (mg-state))
  →  (p-step (p-state (tag ('pc, '(mg-simple-variable-eq . 5)),
                      push (p-frame (list (cons ('ans,
                                                value (car (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk)))),
                                                cons ('x,
                                                      value (cadr (call-actuals (stmt)),
                                                             bindings (top (ctrl-stk)))),
                                                cons ('y,
                                                      value (caddr (call-actuals (stmt)),
                                                             bindings (top (ctrl-stk))))),
                                          tag ('pc,
                                               cons (subr, length (code (cinfo))
                                                           +  4))),
                                    ctrl-stk),
                      push (tag ('bool, eq-value),
                            map-down-values (mg-alist (mg-state),
                                             bindings (top (ctrl-stk)),
                                             temp-stk)),
                      translate-proc-list (proc-list),
                      list (list ('c-c,
                                  mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                      MG-MAX-CTRL-STK-SIZE,
                      MG-MAX-TEMP-STK-SIZE,
                      MG-WORD-SIZE,
                      'run))
  =  p-state (tag ('pc, '(mg-simple-variable-eq . 6)),
              push (p-frame (list (cons ('ans,
                                         value (car (call-actuals (stmt)),
                                                bindings (top (ctrl-stk)))),
                                         cons ('x,
                                               value (cadr (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk)))),
                                         cons ('y,
                                               value (caddr (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk))))),
                                    tag ('pc,
                                         cons (subr, length (code (cinfo))
                                                     +  4))),
                            ctrl-stk),
              push (value (car (call-actuals (stmt)),
                           bindings (top (ctrl-stk))),
                    push (tag ('bool, eq-value),
```

map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-variable-eq-step-11
((*n* ≉ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-variable-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc, '(mg-simple-variable-eq . 6)),
push (p-frame (list (cons ('ans,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('x,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('y,
value (caddr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))))),

```
                                tag('pc,
                                    cons(subr, length(code(cinfo))
                                               +  4))),
                            ctrl-stk),
                        push(value(car(call-actuals(stmt)),
                                   bindings(top(ctrl-stk))),
                            push(tag('bool, eq-value),
                                 map-down-values(mg-alist(mg-state),
                                                 bindings(top(ctrl-stk)),
                                                 temp-stk))),
                        translate-proc-list(proc-list),
                        list(list('c-c,
                                  mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
                        MG-MAX-CTRL-STK-SIZE,
                        MG-MAX-TEMP-STK-SIZE,
                        MG-WORD-SIZE,
                        'run))
=   p-state(tag('pc, '(mg-simple-variable-eq . 7)),
            push(p-frame(list(cons('ans,
                                   value(car(call-actuals(stmt)),
                                         bindings(top(ctrl-stk)))),
                              cons('x,
                                   value(cadr(call-actuals(stmt)),
                                         bindings(top(ctrl-stk)))),
                              cons('y,
                                   value(caddr(call-actuals(stmt)),
                                         bindings(top(ctrl-stk))))),
                         tag('pc,
                             cons(subr, length(code(cinfo))
                                        +  4))),
                    ctrl-stk),
            rput(tag('bool, eq-value),
                 untag(value(car(call-actuals(stmt)),
                             bindings(top(ctrl-stk)))),
                 map-down-values(mg-alist(mg-state),
                                 bindings(top(ctrl-stk)),
                                 temp-stk)),
            translate-proc-list(proc-list),
            list(list('c-c,
                      mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
            MG-MAX-CTRL-STK-SIZE,
            MG-MAX-TEMP-STK-SIZE,
            MG-WORD-SIZE,
            'run))
```

THEOREM: mg-bool-ok-boolean
ok-mg-valuep (mg-bool (*x*), `'boolean-mg`)

THEOREM: mg-simple-variable-eq-step-12-true-case
((*n* ≠ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-variable-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*)
∧ (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (*stmt*)),
mg-alist (*mg-state*))))))
= untag (mg-to-p-simple-literal (caddr (assoc (caddr (call-actuals (*stmt*)),
mg-alist (*mg-state*)))))))
→ (p-step (p-state (tag (`'pc`, `'(mg-simple-variable-eq . 7)`),
push (p-frame (list (cons (`'ans`,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons (`'x`,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons (`'y`,
value (caddr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))))),
tag (`'pc`,
cons (*subr*, length (code (*cinfo*))
+ `4`))),
ctrl-stk),

```
                rput (tag ('bool, 't),
                      untag (value (car (call-actuals (stmt)),
                                        bindings (top (ctrl-stk)))),
                      map-down-values (mg-alist (mg-state),
                                                bindings (top (ctrl-stk)),
                                                temp-stk)),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))
=   p-state (tag ('pc,
                  cons (subr,
                        if normal (mg-meaning-r (stmt,
                                                 proc-list,
                                                 mg-state,
                                                 n,
                                                 list (length (temp-stk),
                                                       p-ctrl-stk-size (ctrl-stk))))
                        then length (code (translate (cinfo,
                                                      t-cond-list,
                                                      stmt,
                                                      proc-list)))
                        else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                        proc-list,
                                                                        mg-state,
                                                                        n,
                                                                        list (length (temp-stk),
                                                                              p-ctrl-stk-size (ctrl-stk)))),
                                                          label-alist (translate (cinfo,
                                                                                  t-cond-list,
                                                                                  stmt,
                                                                                  proc-list))),
                                                     append (code (translate (cinfo,
                                                                              t-cond-list,
                                                                              stmt,
                                                                              proc-list)),
                                                             code2)) endif)),
                  ctrl-stk,
                  map-down-values (mg-alist (mg-meaning-r (stmt,
                                                          proc-list,
                                                          mg-state,
```

n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-variable-eq-step-12-false-case
((*n* $\not\simeq$ 0)
$\wedge$ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
$\wedge$ (car (*stmt*) = `'predefined-proc-call-mg`)
$\wedge$ (call-name (*stmt*) = `'mg-simple-variable-eq`)
$\wedge$ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
$\wedge$ ok-mg-def-plistp (*proc-list*)
$\wedge$ ok-mg-statep (*mg-state*, *r-cond-list*)
$\wedge$ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
$\wedge$ user-defined-procp (*subr*, *proc-list*)
$\wedge$ listp (*ctrl-stk*)
$\wedge$ all-cars-unique (mg-alist (*mg-state*))
$\wedge$ signatures-match (mg-alist (*mg-state*), *name-alist*)
$\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
$\wedge$ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
$\wedge$ normal (*mg-state*)
$\wedge$ (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (*stmt*)),
mg-alist (*mg-state*))))))

```
    ≠  untag(mg-to-p-simple-literal(caddr(assoc(caddr(call-actuals(stmt)),
                                                mg-alist(mg-state)))))))
→   (p-step(p-state(tag('pc, '(mg-simple-variable-eq . 7)),
                   push(p-frame(list(cons('ans,
                                          value(car(call-actuals(stmt)),
                                                bindings(top(ctrl-stk)))),
                                     cons('x,
                                          value(cadr(call-actuals(stmt)),
                                                bindings(top(ctrl-stk)))),
                                     cons('y,
                                          value(caddr(call-actuals(stmt)),
                                                bindings(top(ctrl-stk))))),
                                tag('pc,
                                    cons(subr, length(code(cinfo))
                                               + 4))),
                        ctrl-stk),
                   rput(tag('bool, 'f),
                        untag(value(car(call-actuals(stmt)),
                                    bindings(top(ctrl-stk)))),
                        map-down-values(mg-alist(mg-state),
                                        bindings(top(ctrl-stk)),
                                        temp-stk)),
                   translate-proc-list(proc-list),
                   list(list('c-c,
                             mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
                   MG-MAX-CTRL-STK-SIZE,
                   MG-MAX-TEMP-STK-SIZE,
                   MG-WORD-SIZE,
                   'run))
    =  p-state(tag('pc,
                   cons(subr,
                        if normal(mg-meaning-r(stmt,
                                               proc-list,
                                               mg-state,
                                               n,
                                               list(length(temp-stk),
                                                    p-ctrl-stk-size(ctrl-stk))))
                        then length(code(translate(cinfo,
                                                   t-cond-list,
                                                   stmt,
                                                   proc-list)))
                        else find-label(fetch-label(cc(mg-meaning-r(stmt,
                                                                    proc-list,
                                                                    mg-state,
```

n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
label-alist (translate (*cinfo*,
t-cond-list,
stmt,
proc-list))),
append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-variable-eq-exact-time-lemma
(($n \not\simeq$ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-variable-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)

```
∧  ok-mg-def-plistp (proc-list)
∧  ok-mg-statep (mg-state, r-cond-list)
∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
    =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                    code2))
∧  user-defined-procp (subr, proc-list)
∧  listp (ctrl-stk)
∧  all-cars-unique (mg-alist (mg-state))
∧  signatures-match (mg-alist (mg-state), name-alist)
∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
∧  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
∧  normal (mg-state))
→  (p (map-down (mg-state,
                 proc-list,
                 ctrl-stk,
                 temp-stk,
                 tag ('pc, cons (subr, length (code (cinfo)))),
                 t-cond-list),
       clock (stmt, proc-list, mg-state, n))
    =  p-state (tag ('pc,
                     cons (subr,
                           if normal (mg-meaning-r (stmt,
                                                    proc-list,
                                                    mg-state,
                                                    n,
                                                    list (length (temp-stk),
                                                          p-ctrl-stk-size (ctrl-stk))))
                           then length (code (translate (cinfo,
                                                         t-cond-list,
                                                         stmt,
                                                         proc-list)))
                           else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                           proc-list,
                                                                           mg-state,
                                                                           n,
                                                                           list (length (temp-stk),
                                                                                 p-ctrl-stk-size (ctrl-stk)))),
                                                             label-alist (translate (cinfo,
                                                                                     t-cond-list,
                                                                                     stmt,
                                                                                     proc-list))),
                                                        append (code (translate (cinfo,
```

```
                                                          t-cond-list,
                                                          stmt,
                                                          proc-list)),
                                              code2)) endif)),
                    ctrl-stk,
                    map-down-values (mg-alist (mg-meaning-r (stmt,
                                                             proc-list,
                                                             mg-state,
                                                             n,
                                                             list (length (temp-stk),
                                                                   p-ctrl-stk-size (ctrl-stk)))),
                                    bindings (top (ctrl-stk)),
                                    temp-stk),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                               mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                                   proc-list,
                                                                   mg-state,
                                                                   n,
                                                                   list (length (temp-stk),
                                                                         p-ctrl-stk-size (ctrl-stk)))),
                                                 t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                                                                          ;;
;;                 EXACT-TIME LEMMA MG-SIMPLE-CONSTANT-EQ                   ;;
;;                                                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

THEOREM: mg-simple-constant-eq-args-simple-identifierps

```
 (ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
  ∧   (car (stmt) = 'predefined-proc-call-mg)
  ∧   (call-name (stmt) = 'mg-simple-constant-eq)
  ∧   ok-mg-statep (mg-state, r-cond-list)
  ∧   signatures-match (mg-alist (mg-state), name-alist))
 →    (boolean-identifierp (car (call-actuals (stmt)), mg-alist (mg-state))
        ∧   simple-identifierp (cadr (call-actuals (stmt)),
                                mg-alist (mg-state))
        ∧   simple-typed-literalp (caddr (call-actuals (stmt)),
```

cadr (assoc (cadr (call-actuals (*stmt*)),
mg-alist (*mg-state*)))))

THEOREM: mg-simple-constant-eq-args-definedp
(ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-eq`)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ signatures-match (mg-alist (*mg-state*), *name-alist*))
→ (definedp (car (call-actuals (*stmt*)), mg-alist (*mg-state*))
∧ definedp (cadr (call-actuals (*stmt*)), mg-alist (*mg-state*)))

THEOREM: mg-simple-constant-eq-steps-1-3
((*n* ≠ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-step (p-step (map-down (*mg-state*,
proc-list,
ctrl-stk,
temp-stk,
tag (`'pc`,
cons (*subr*, length (code (*cinfo*)))),
t-cond-list))))
= p-state (tag (`'pc`, cons (*subr*, length (code (*cinfo*)) + `3`)),
ctrl-stk,

```
                push (mg-to-p-simple-literal (caddr (call-actuals (stmt))),
                      push (value (cadr (call-actuals (stmt)),
                                  bindings (top (ctrl-stk))),
                           push (value (car (call-actuals (stmt)),
                                       bindings (top (ctrl-stk))),
                                map-down-values (mg-alist (mg-state),
                                                 bindings (top (ctrl-stk)),
                                                 temp-stk)))),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))
```

THEOREM: mg-simple-constant-eq-step-4

```
((n ≠ 0)
 ∧  (¬ resources-inadequatep (stmt,
                              proc-list,
                              list (length (temp-stk),
                                    p-ctrl-stk-size (ctrl-stk))))
 ∧  (car (stmt) = 'predefined-proc-call-mg)
 ∧  (call-name (stmt) = 'mg-simple-constant-eq)
 ∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
 ∧  ok-mg-def-plistp (proc-list)
 ∧  ok-mg-statep (mg-state, r-cond-list)
 ∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                code2))
 ∧  user-defined-procp (subr, proc-list)
 ∧  listp (ctrl-stk)
 ∧  all-cars-unique (mg-alist (mg-state))
 ∧  signatures-match (mg-alist (mg-state), name-alist)
 ∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                                bindings (top (ctrl-stk)),
                                temp-stk)
 ∧  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
 ∧  normal (mg-state))
→  (p-step (p-state (tag ('pc, cons (subr, length (code (cinfo)) + 3)),
                     ctrl-stk,
                     push (mg-to-p-simple-literal (caddr (call-actuals (stmt))),
                           push (value (cadr (call-actuals (stmt)),
                                       bindings (top (ctrl-stk))),
```

```
                                    push (value (car (call-actuals (stmt)),
                                                     bindings (top (ctrl-stk))),
                                         map-down-values (mg-alist (mg-state),
                                                                   bindings (top (ctrl-stk)),
                                                                   temp-stk))))),
                               translate-proc-list (proc-list),
                               list (list ('c-c,
                                           mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                               MG-MAX-CTRL-STK-SIZE,
                               MG-MAX-TEMP-STK-SIZE,
                               MG-WORD-SIZE,
                               'run))
    =    p-state (tag ('pc, '(mg-simple-constant-eq . 0)),
                  push (p-frame (list (cons ('ans,
                                             value (car (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk))))),
                                      cons ('x,
                                             value (cadr (call-actuals (stmt)),
                                                    bindings (top (ctrl-stk))))),
                                      cons ('y,
                                             mg-to-p-simple-literal (caddr (call-actuals (stmt)))))),
                                tag ('pc,
                                     cons (subr, length (code (cinfo))
                                                 +    4))),
                        ctrl-stk),
                  map-down-values (mg-alist (mg-state),
                                             bindings (top (ctrl-stk)),
                                             temp-stk),
                  translate-proc-list (proc-list),
                  list (list ('c-c,
                              mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                  MG-MAX-CTRL-STK-SIZE,
                  MG-MAX-TEMP-STK-SIZE,
                  MG-WORD-SIZE,
                  'run))
```

THEOREM: mg-simple-constant-eq-step-5

```
((n ≠ 0)
 ∧   (¬ resources-inadequatep (stmt,
                               proc-list,
                               list (length (temp-stk),
                                     p-ctrl-stk-size (ctrl-stk))))
 ∧   (car (stmt) = 'predefined-proc-call-mg)
 ∧   (call-name (stmt) = 'mg-simple-constant-eq)
```

```
∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
∧  ok-mg-def-plistp (proc-list)
∧  ok-mg-statep (mg-state, r-cond-list)
∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                     code2))
∧  user-defined-procp (subr, proc-list)
∧  listp (ctrl-stk)
∧  all-cars-unique (mg-alist (mg-state))
∧  signatures-match (mg-alist (mg-state), name-alist)
∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
∧  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
∧  normal (mg-state))
→  (p-step (p-state (tag ('pc, '(mg-simple-constant-eq . 0)),
                    push (p-frame (list (cons ('ans,
                                               value (car (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk)))),
                                          cons ('x,
                                                value (cadr (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk)))),
                                          cons ('y,
                                                mg-to-p-simple-literal (caddr (call-actuals (stmt))))),
                                     tag ('pc,
                                          cons (subr, length (code (cinfo))
                                                      +  4))),
                          ctrl-stk),
                    map-down-values (mg-alist (mg-state),
                                     bindings (top (ctrl-stk)),
                                     temp-stk),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
     =  p-state (tag ('pc, '(mg-simple-constant-eq . 1)),
                 push (p-frame (list (cons ('ans,
                                            value (car (call-actuals (stmt)),
                                                   bindings (top (ctrl-stk)))),
                                       cons ('x,
                                             value (cadr (call-actuals (stmt)),
```

bindings (top (*ctrl-stk*)))),
cons (`'y`,
mg-to-p-simple-literal (caddr (call-actuals (*stmt*)))))),
tag (`'pc`,
cons (*subr*, length (code (*cinfo*))
+ `4`))),
ctrl-stk),
push (value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-constant-eq-step-6
((*n* $\not\simeq$ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag (`'pc`, `'(mg-simple-constant-eq . 1)`)),

```
                push (p-frame (list (cons ('ans,
                                           value (car (call-actuals (stmt)),
                                                  bindings (top (ctrl-stk)))),
                                    cons ('x,
                                          value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk)))),
                                    cons ('y,
                                          mg-to-p-simple-literal (caddr (call-actuals (stmt))))),
                              tag ('pc,
                                   cons (subr, length (code (cinfo))
                                               + 4))),
                      ctrl-stk),
                push (value (cadr (call-actuals (stmt)),
                            bindings (top (ctrl-stk))),
                      map-down-values (mg-alist (mg-state),
                                       bindings (top (ctrl-stk)),
                                       temp-stk)),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))
=   p-state (tag ('pc, '(mg-simple-constant-eq . 2)),
                push (p-frame (list (cons ('ans,
                                           value (car (call-actuals (stmt)),
                                                  bindings (top (ctrl-stk)))),
                                    cons ('x,
                                          value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk)))),
                                    cons ('y,
                                          mg-to-p-simple-literal (caddr (call-actuals (stmt))))),
                              tag ('pc,
                                   cons (subr, length (code (cinfo))
                                               + 4))),
                      ctrl-stk),
                push (rget (untag (value (cadr (call-actuals (stmt)),
                                         bindings (top (ctrl-stk)))),
                           push (value (cadr (call-actuals (stmt)),
                                       bindings (top (ctrl-stk))),
                                 map-down-values (mg-alist (mg-state),
                                                  bindings (top (ctrl-stk)),
                                                  temp-stk))),
```

map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-constant-eq-step-7
((*n* ≉ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-constant-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc, '(mg-simple-constant-eq . 2)),
push (p-frame (list (cons ('ans,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('x,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('y,
mg-to-p-simple-literal (caddr (call-actuals (*stmt*))))),
tag ('pc,

```
                                    cons (subr, length (code (cinfo))
                                                + 4))),
                            ctrl-stk),
                        push (rget (untag (value (cadr (call-actuals (stmt)),
                                                 bindings (top (ctrl-stk)))),
                                   push (value (cadr (call-actuals (stmt)),
                                                     bindings (top (ctrl-stk))),
                                         map-down-values (mg-alist (mg-state),
                                                          bindings (top (ctrl-stk)),
                                                          temp-stk))),
                                   map-down-values (mg-alist (mg-state),
                                                    bindings (top (ctrl-stk)),
                                                    temp-stk)),
                        translate-proc-list (proc-list),
                        list (list ('c-c,
                                    mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                        MG-MAX-CTRL-STK-SIZE,
                        MG-MAX-TEMP-STK-SIZE,
                        MG-WORD-SIZE,
                        'run))
=   p-state (tag ('pc, '(mg-simple-constant-eq . 3)),
                push (p-frame (list (cons ('ans,
                                           value (car (call-actuals (stmt)),
                                                  bindings (top (ctrl-stk)))),
                                          cons ('x,
                                                value (cadr (call-actuals (stmt)),
                                                       bindings (top (ctrl-stk)))),
                                          cons ('y,
                                                mg-to-p-simple-literal (caddr (call-actuals (stmt))))),
                                    tag ('pc,
                                         cons (subr, length (code (cinfo))
                                                     + 4))),
                            ctrl-stk),
                      push (mg-to-p-simple-literal (caddr (call-actuals (stmt))),
                            push (rget (untag (value (cadr (call-actuals (stmt)),
                                                     bindings (top (ctrl-stk)))),
                                       push (value (cadr (call-actuals (stmt)),
                                                   bindings (top (ctrl-stk))),
                                             map-down-values (mg-alist (mg-state),
                                                              bindings (top (ctrl-stk)),
                                                              temp-stk))),
                                       map-down-values (mg-alist (mg-state),
                                                        bindings (top (ctrl-stk)),
                                                        temp-stk))),
```

translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-constant-eq-step-8
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-constant-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc, '(mg-simple-constant-eq . 3)),
push (p-frame (list (cons ('ans,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('x,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('y,
mg-to-p-simple-literal (caddr (call-actuals (*stmt*))))),
tag ('pc,
cons (*subr*, length (code (*cinfo*))
+ 4))),
ctrl-stk),

```
                push (mg-to-p-simple-literal (caddr (call-actuals (stmt))),
                      push (rget (untag (value (cadr (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                 push (value (cadr (call-actuals (stmt)),
                                             bindings (top (ctrl-stk))),
                                       map-down-values (mg-alist (mg-state),
                                                        bindings (top (ctrl-stk)),
                                                        temp-stk))),
                           map-down-values (mg-alist (mg-state),
                                            bindings (top (ctrl-stk)),
                                            temp-stk))),
                translate-proc-list (proc-list),
                list (list ('c-c,
                            mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                MG-MAX-CTRL-STK-SIZE,
                MG-MAX-TEMP-STK-SIZE,
                MG-WORD-SIZE,
                'run))
=   p-state (tag ('pc, '(mg-simple-constant-eq . 4)),
             push (p-frame (list (cons ('ans,
                                        value (car (call-actuals (stmt)),
                                               bindings (top (ctrl-stk)))),
                                 cons ('x,
                                       value (cadr (call-actuals (stmt)),
                                              bindings (top (ctrl-stk)))),
                                 cons ('y,
                                       mg-to-p-simple-literal (caddr (call-actuals (stmt))))),
                           tag ('pc,
                                cons (subr, length (code (cinfo))
                                           +  4))),
                   ctrl-stk),
             push (tag ('bool,
                        if untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                                       mg-alist (mg-state)))))
                          =  untag (mg-to-p-simple-literal (caddr (call-actuals (stmt))))
                        then 't
                        else 'f endif),
                   map-down-values (mg-alist (mg-state),
                                    bindings (top (ctrl-stk)),
                                    temp-stk)),
             translate-proc-list (proc-list),
             list (list ('c-c,
                         mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
             MG-MAX-CTRL-STK-SIZE,
```

MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-constant-eq-step-9
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-constant-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p-step (p-state (tag ('pc, '(mg-simple-constant-eq . 4)),
push (p-frame (list (cons ('ans,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('x,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('y,
mg-to-p-simple-literal (caddr (call-actuals (*stmt*))))),
tag ('pc,
cons (*subr*, length (code (*cinfo*))
+ 4))),
ctrl-stk),
push (tag ('bool, *eq-value*),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),

translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

= p-state (tag ('pc, '(mg-simple-constant-eq . 5)),
push (p-frame (list (cons ('ans,
value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('x,
value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons ('y,
mg-to-p-simple-literal (caddr (call-actuals (*stmt*))))),
tag ('pc,
cons (*subr*, length (code (*cinfo*))
+ 4))),
ctrl-stk),
push (value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*))),
push (tag ('bool, *eq-value*),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk))),
translate-proc-list (*proc-list*),
list (list ('c-c,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
'run))

THEOREM: mg-simple-constant-eq-step-10
((*n* ≠ 0)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = 'predefined-proc-call-mg)
∧ (call-name (*stmt*) = 'mg-simple-constant-eq)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)

```
∧  ok-mg-statep(mg-state, r-cond-list)
∧  (code(translate-def-body(assoc(subr, proc-list), proc-list))
     =  append(code(translate(cinfo, t-cond-list, stmt, proc-list)),
               code2))
∧  user-defined-procp(subr, proc-list)
∧  listp(ctrl-stk)
∧  all-cars-unique(mg-alist(mg-state))
∧  signatures-match(mg-alist(mg-state), name-alist)
∧  mg-vars-list-ok-in-p-state(mg-alist(mg-state),
                              bindings(top(ctrl-stk)),
                              temp-stk)
∧  no-p-aliasing(bindings(top(ctrl-stk)), mg-alist(mg-state))
∧  normal(mg-state))
→  (p-step(p-state(tag('pc, '(mg-simple-constant-eq . 5)),
                   push(p-frame(list(cons('ans,
                                          value(car(call-actuals(stmt)),
                                                bindings(top(ctrl-stk))))),
                                     cons('x,
                                          value(cadr(call-actuals(stmt)),
                                                bindings(top(ctrl-stk))))),
                                     cons('y,
                                          mg-to-p-simple-literal(caddr(call-actuals(stmt))))),
                              tag('pc,
                                  cons(subr, length(code(cinfo))
                                             +  4))),
                        ctrl-stk),
                   push(value(car(call-actuals(stmt)),
                              bindings(top(ctrl-stk))),
                        push(tag('bool, eq-value),
                             map-down-values(mg-alist(mg-state),
                                             bindings(top(ctrl-stk)),
                                             temp-stk))),
                   translate-proc-list(proc-list),
                   list(list('c-c,
                             mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
                   MG-MAX-CTRL-STK-SIZE,
                   MG-MAX-TEMP-STK-SIZE,
                   MG-WORD-SIZE,
                   'run))
     =  p-state(tag('pc, '(mg-simple-constant-eq . 6)),
                push(p-frame(list(cons('ans,
                                       value(car(call-actuals(stmt)),
                                             bindings(top(ctrl-stk))))),
                                  cons('x,
```

value (cadr (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
cons (`'y`,
mg-to-p-simple-literal (caddr (call-actuals (*stmt*))))),
tag (`'pc`,
cons (*subr*, length (code (*cinfo*))
+ `4`))),
ctrl-stk),
rput (tag (`'bool`, *eq-value*),
untag (value (car (call-actuals (*stmt*)),
bindings (top (*ctrl-stk*)))),
map-down-values (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (*mg-state*), *t-cond-list*))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-constant-eq-step-11-true-case
((*n* ≉ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))

```
  ∧  normal(mg-state)
  ∧  (untag(mg-to-p-simple-literal(caddr(assoc(cadr(call-actuals(stmt)),
                                               mg-alist(mg-state))))))
     =  untag(mg-to-p-simple-literal(caddr(call-actuals(stmt))))))
  →  (p-step(p-state(tag('pc, '(mg-simple-constant-eq . 6)),
                     push(p-frame(list(cons('ans,
                                            value(car(call-actuals(stmt)),
                                                  bindings(top(ctrl-stk)))),
                                       cons('x,
                                            value(cadr(call-actuals(stmt)),
                                                  bindings(top(ctrl-stk)))),
                                       cons('y,
                                            mg-to-p-simple-literal(caddr(call-actuals(stmt))))),
                                  tag('pc,
                                      cons(subr, length(code(cinfo))
                                                 +  4))),
                          ctrl-stk),
                     rput(tag('bool, 't),
                          untag(value(car(call-actuals(stmt)),
                                      bindings(top(ctrl-stk)))),
                          map-down-values(mg-alist(mg-state),
                                          bindings(top(ctrl-stk)),
                                          temp-stk)),
                     translate-proc-list(proc-list),
                     list(list('c-c,
                               mg-cond-to-p-nat(cc(mg-state), t-cond-list))),
                     MG-MAX-CTRL-STK-SIZE,
                     MG-MAX-TEMP-STK-SIZE,
                     MG-WORD-SIZE,
                     'run))
     =  p-state(tag('pc,
                    cons(subr,
                         if normal(mg-meaning-r(stmt,
                                                proc-list,
                                                mg-state,
                                                n,
                                                list(length(temp-stk),
                                                     p-ctrl-stk-size(ctrl-stk))))
                         then length(code(translate(cinfo,
                                                    t-cond-list,
                                                    stmt,
                                                    proc-list)))
                         else find-label(fetch-label(cc(mg-meaning-r(stmt,
                                                                    proc-list,
```

mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
label-alist (translate (*cinfo*,
t-cond-list,
stmt,
proc-list))),
append (code (translate (*cinfo*,
t-cond-list,
stmt,
proc-list)),
code2)) **endif**)),
ctrl-stk,
map-down-values (mg-alist (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
bindings (top (*ctrl-stk*)),
temp-stk),
translate-proc-list (*proc-list*),
list (list (`'c-c`,
mg-cond-to-p-nat (cc (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*)))),
t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-constant-eq-step-11-false-case
((*n* ≠ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-eq`)

```
∧  ok-mg-statement (stmt, r-cond-list, name-alist, proc-list)
∧  ok-mg-def-plistp (proc-list)
∧  ok-mg-statep (mg-state, r-cond-list)
∧  (code (translate-def-body (assoc (subr, proc-list), proc-list))
     =  append (code (translate (cinfo, t-cond-list, stmt, proc-list)),
                     code2))
∧  user-defined-procp (subr, proc-list)
∧  listp (ctrl-stk)
∧  all-cars-unique (mg-alist (mg-state))
∧  signatures-match (mg-alist (mg-state), name-alist)
∧  mg-vars-list-ok-in-p-state (mg-alist (mg-state),
                               bindings (top (ctrl-stk)),
                               temp-stk)
∧  no-p-aliasing (bindings (top (ctrl-stk)), mg-alist (mg-state))
∧  normal (mg-state)
∧  (untag (mg-to-p-simple-literal (caddr (assoc (cadr (call-actuals (stmt)),
                                                mg-alist (mg-state))))))
     ≠  untag (mg-to-p-simple-literal (caddr (call-actuals (stmt)))))))
→  (p-step (p-state (tag ('pc, '(mg-simple-constant-eq . 6)),
                    push (p-frame (list (cons ('ans,
                                               value (car (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk)))),
                                          cons ('x,
                                               value (cadr (call-actuals (stmt)),
                                                      bindings (top (ctrl-stk)))),
                                          cons ('y,
                                               mg-to-p-simple-literal (caddr (call-actuals (stmt))))),
                                  tag ('pc,
                                       cons (subr, length (code (cinfo))
                                                   +  4))),
                          ctrl-stk),
                    rput (tag ('bool, 'f),
                          untag (value (car (call-actuals (stmt)),
                                            bindings (top (ctrl-stk)))),
                          map-down-values (mg-alist (mg-state),
                                           bindings (top (ctrl-stk)),
                                           temp-stk)),
                    translate-proc-list (proc-list),
                    list (list ('c-c,
                                mg-cond-to-p-nat (cc (mg-state), t-cond-list))),
                    MG-MAX-CTRL-STK-SIZE,
                    MG-MAX-TEMP-STK-SIZE,
                    MG-WORD-SIZE,
                    'run))
```

```
=  p-state (tag ('pc,
             cons (subr,
                   if normal (mg-meaning-r (stmt,
                                            proc-list,
                                            mg-state,
                                            n,
                                            list (length (temp-stk),
                                                  p-ctrl-stk-size (ctrl-stk))))
                   then length (code (translate (cinfo,
                                                 t-cond-list,
                                                 stmt,
                                                 proc-list)))
                   else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                   proc-list,
                                                                   mg-state,
                                                                   n,
                                                                   list (length (temp-stk),
                                                                         p-ctrl-stk-size (ctrl-stk)))),
                                                     label-alist (translate (cinfo,
                                                                             t-cond-list,
                                                                             stmt,
                                                                             proc-list))),
                                              append (code (translate (cinfo,
                                                                       t-cond-list,
                                                                       stmt,
                                                                       proc-list)),
                                                      code2)) endif)),
             ctrl-stk,
             map-down-values (mg-alist (mg-meaning-r (stmt,
                                                      proc-list,
                                                      mg-state,
                                                      n,
                                                      list (length (temp-stk),
                                                            p-ctrl-stk-size (ctrl-stk)))),
                              bindings (top (ctrl-stk)),
                              temp-stk),
             translate-proc-list (proc-list),
             list (list ('c-c,
                         mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                            proc-list,
                                                            mg-state,
                                                            n,
                                                            list (length (temp-stk),
                                                                  p-ctrl-stk-size (ctrl-stk)))),
```

t-cond-list))),
MG-MAX-CTRL-STK-SIZE,
MG-MAX-TEMP-STK-SIZE,
MG-WORD-SIZE,
`'run`))

THEOREM: mg-simple-constant-eq-exact-time-lemma
((*n* ≠ `0`)
∧ (¬ resources-inadequatep (*stmt*,
proc-list,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))
∧ (car (*stmt*) = `'predefined-proc-call-mg`)
∧ (call-name (*stmt*) = `'mg-simple-constant-eq`)
∧ ok-mg-statement (*stmt*, *r-cond-list*, *name-alist*, *proc-list*)
∧ ok-mg-def-plistp (*proc-list*)
∧ ok-mg-statep (*mg-state*, *r-cond-list*)
∧ (code (translate-def-body (assoc (*subr*, *proc-list*), *proc-list*))
= append (code (translate (*cinfo*, *t-cond-list*, *stmt*, *proc-list*)),
code2))
∧ user-defined-procp (*subr*, *proc-list*)
∧ listp (*ctrl-stk*)
∧ all-cars-unique (mg-alist (*mg-state*))
∧ signatures-match (mg-alist (*mg-state*), *name-alist*)
∧ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
bindings (top (*ctrl-stk*)),
temp-stk)
∧ no-p-aliasing (bindings (top (*ctrl-stk*)), mg-alist (*mg-state*))
∧ normal (*mg-state*))
→ (p (map-down (*mg-state*,
proc-list,
ctrl-stk,
temp-stk,
tag (`'pc`, cons (*subr*, length (code (*cinfo*)))),
t-cond-list),
clock (*stmt*, *proc-list*, *mg-state*, *n*))
= p-state (tag (`'pc`,
cons (*subr*,
if normal (mg-meaning-r (*stmt*,
proc-list,
mg-state,
n,
list (length (*temp-stk*),
p-ctrl-stk-size (*ctrl-stk*))))

```
                then length (code (translate (cinfo,
                                              t-cond-list,
                                              stmt,
                                              proc-list)))
                else find-label (fetch-label (cc (mg-meaning-r (stmt,
                                                                proc-list,
                                                                mg-state,
                                                                n,
                                                                list (length (temp-stk),
                                                                      p-ctrl-stk-size (ctrl-stk)))),
                                                  label-alist (translate (cinfo,
                                                                          t-cond-list,
                                                                          stmt,
                                                                          proc-list))),
                                                  append (code (translate (cinfo,
                                                                           t-cond-list,
                                                                           stmt,
                                                                           proc-list)),
                                                          code2)) endif)),
            ctrl-stk,
            map-down-values (mg-alist (mg-meaning-r (stmt,
                                                     proc-list,
                                                     mg-state,
                                                     n,
                                                     list (length (temp-stk),
                                                           p-ctrl-stk-size (ctrl-stk)))),
                             bindings (top (ctrl-stk)),
                             temp-stk),
            translate-proc-list (proc-list),
            list (list ('c-c,
                        mg-cond-to-p-nat (cc (mg-meaning-r (stmt,
                                                           proc-list,
                                                           mg-state,
                                                           n,
                                                           list (length (temp-stk),
                                                                 p-ctrl-stk-size (ctrl-stk)))),
                                          t-cond-list))),
            MG-MAX-CTRL-STK-SIZE,
            MG-MAX-TEMP-STK-SIZE,
            MG-WORD-SIZE,
            'run))
```

EVENT: Make the library `"c-predefined1"`.

Index