

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 illessp.

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 $n \simeq 0$ then signal-system-error (mg-state, 'timed-out)
elseif $\neg$ 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-literal (*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*)))
= illessp (untag (*x*), untag (*y*)))

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

THEOREM: alist-element-type-conversion
 (mg-alistp (*lst*) $\wedge$ simple-identifierp (*x*, *lst*))
 $\rightarrow$ (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*)
 $\rightarrow$ (type (mg-to-p-simple-literal (*lit*))
 = **if** *type* = 'boolean-mg **then** 'bool
 else 'int **endif**)

THEOREM: int-literals-mapping
 int-literalp (*x*)
 $\rightarrow$ ((type (mg-to-p-simple-literal (*x*)) = 'int)
 $\wedge$ listp (mg-to-p-simple-literal (*x*))
 $\wedge$ (caddr (mg-to-p-simple-literal (*x*)) = **nil**)
 $\wedge$ small-integerp (untag (mg-to-p-simple-literal (*x*)), 32))

THEOREM: int-identifiers-have-int-literal-values
 (mg-alistp (*mg-vars*) $\wedge$ int-identifierp (*x*, *mg-vars*))
 $\rightarrow$ int-literalp (caddr (assoc (*x*, *mg-vars*)))

THEOREM: boolean-identifiers-have-boolean-literal-values
 (mg-alistp (*mg-vars*) $\wedge$ boolean-identifierp (*x*, *mg-vars*))
 $\rightarrow$ boolean-literalp (caddr (assoc (*x*, *mg-vars*)))

THEOREM: length-plistp-2-rewrite
 (caddr (*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*)
 $\wedge$ simple-identifierp (*b*, *mg-vars*)
 $\wedge$ (length (*temp-stk*) < MG-MAX-TEMP-STK-SIZE)
 $\wedge$ all-cars-unique (*mg-vars*)
 $\rightarrow$ ((type (value (*b*, *bindings*)) = 'nat)
 $\wedge$ (caddr (value (*b*, *bindings*)) = **nil**)
 $\wedge$ listp (value (*b*, *bindings*))
 $\wedge$ 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)
 ∧ (caddr (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
 $\text{int-literalp}(x) \rightarrow \text{small-integerp}(\text{untag}(x), 32)$

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

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

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

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

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

THEOREM: bool-literal-bool-objectp
 $\text{p-objectp-type}(' \text{bool}, '(\text{bool } t), \text{state})$
 $\wedge \text{p-objectp-type}(' \text{bool}, '(\text{bool } f), \text{state})$

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

```

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

```

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

THEOREM: mg-simple-variable-assignment-args-simple-identifierps
 (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*)
 $\rightarrow$ (simple-identifierp (car (call-actuals (*stmt*))), mg-alist (*mg-state*))
 $\wedge$ simple-identifierp (cadr (call-actuals (*stmt*))),
 mg-alist (*mg-state*)))

THEOREM: mg-simple-variable-assignment-steps-1-2
 (($\neg$ 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-assignment)
 $\wedge$ 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*)
 $\wedge$ mg-vars-list-ok-in-p-state (mg-alist (*mg-state*),
 bindings (top (*ctrl-stk*)),
temp-stk)
 $\wedge$ normal (*mg-state*)
 $\rightarrow$ (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

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

```

      '(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
((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-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-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))
= 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))

```

```

 $\wedge$  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))),
                                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)
 $\wedge$ (car (stmt) = 'predefined-proc-call-mg)
 $\wedge$ (call-name (stmt) = 'mg-simple-constant-assignment)
 $\wedge$ ok-mg-statep (mg-state, r-cond-list)
 $\wedge$ signatures-match (mg-alist (mg-state), name-alist))
 $\rightarrow$ 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),
              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))

```

```

      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)

```

[illegible]

```

                                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*))

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

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

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

```

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

$((n \neq 0)$
 $\wedge (\neg \text{resources-inadequatep} (stmt,$
 $proc-list,$

```

                                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

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

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

```

      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)))),
                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))

```

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

```

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

$((n \neq 0)$
 $\wedge (\neg \text{resources-inadequatep} (stmt,$
 $\quad \quad \quad \text{proc-list},$
 $\quad \quad \quad \text{list} (\text{length} (temp-stk),$
 $\quad \quad \quad \text{p-ctrl-stk-size} (ctrl-stk))))$
 $\wedge (\text{car} (stmt) = \text{'predefined-proc-call-mg})$
 $\wedge (\text{call-name} (stmt) = \text{'mg-simple-variable-eq})$
 $\wedge \text{ok-mg-statement} (stmt, r-cond-list, name-alist, proc-list)$
 $\wedge \text{ok-mg-def-plistp} (proc-list)$
 $\wedge \text{ok-mg-statep} (mg-state, r-cond-list)$
 $\wedge (\text{code} (\text{translate-def-body} (\text{assoc} (subr, proc-list), proc-list))$
 $\quad = \text{append} (\text{code} (\text{translate} (cinfo, t-cond-list, stmt, proc-list)),$
 $\quad \quad \quad \text{code2}))$
 $\wedge \text{user-defined-procp} (subr, proc-list)$
 $\wedge \text{listp} (ctrl-stk)$
 $\wedge \text{all-cars-unique} (\text{mg-alist} (mg-state))$
 $\wedge \text{signatures-match} (\text{mg-alist} (mg-state), name-alist)$
 $\wedge \text{mg-vars-list-ok-in-p-state} (\text{mg-alist} (mg-state),$
 $\quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)),$
 $\quad \quad \quad temp-stk)$
 $\wedge \text{no-p-aliasing} (\text{bindings} (\text{top} (ctrl-stk)), \text{mg-alist} (mg-state))$
 $\wedge \text{normal} (mg-state)$
 $\wedge (\text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{cadr} (\text{call-actuals} (stmt)),$
 $\quad \quad \quad \text{mg-alist} (mg-state))))))$
 $\quad = \text{untag} (\text{mg-to-p-simple-literal} (\text{caddr} (\text{assoc} (\text{caddr} (\text{call-actuals} (stmt)),$
 $\quad \quad \quad \text{mg-alist} (mg-state))))))$
 $\rightarrow (\text{p-step} (\text{p-state} (\text{tag} (\text{'pc}, \text{'(mg-simple-variable-eq . 7)}),$
 $\quad \quad \quad \text{push} (\text{p-frame} (\text{list} (\text{cons} (\text{'ans},$
 $\quad \quad \quad \quad \quad \quad \text{value} (\text{car} (\text{call-actuals} (stmt)),$
 $\quad \quad \quad \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)))),$
 $\quad \quad \quad \text{cons} (\text{'x},$
 $\quad \quad \quad \quad \quad \quad \text{value} (\text{cadr} (\text{call-actuals} (stmt)),$
 $\quad \quad \quad \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk)))),$
 $\quad \quad \quad \text{cons} (\text{'y},$
 $\quad \quad \quad \quad \quad \quad \text{value} (\text{caddr} (\text{call-actuals} (stmt)),$
 $\quad \quad \quad \quad \quad \quad \text{bindings} (\text{top} (ctrl-stk))))),$
 $\quad \quad \quad \text{tag} (\text{'pc},$
 $\quad \quad \quad \quad \quad \quad \text{cons} (subr, \text{length} (\text{code} (cinfo))$
 $\quad \quad \quad \quad \quad \quad + 4))),$
 $\quad \quad \quad ctrl-stk),$

```

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 ≠ 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, '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 ≠ 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)),

```

$$\text{cadr}(\text{assoc}(\text{cadr}(\text{call-actuals}(stmt)), \\ \text{mg-alist}(mg\text{-state}))))))$$

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

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

```

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 ≠ 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

(($n \neq 0$)
 $\wedge$ ($\neg$ 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-constant-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-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))))),

```

$t\text{-cond-list}))$),
 MG-MAX-CTRL-STK-SIZE,
 MG-MAX-TEMP-STK-SIZE,
 MG-WORD-SIZE,
 'run))

THEOREM: mg-simple-constant-eq-exact-time-lemma

$((n \neq 0)$
 $\wedge (\neg \text{resources-inadequatep} (stmt,$
 $\quad proc\text{-list},$
 $\quad \text{list} (\text{length} (temp\text{-stk}),$
 $\quad \text{p-ctrl-stk-size} (ctrl\text{-stk}))))$
 $\wedge (\text{car} (stmt) = \text{'predefined-proc-call-mg})$
 $\wedge (\text{call-name} (stmt) = \text{'mg-simple-constant-eq})$
 $\wedge \text{ok-mg-statement} (stmt, r\text{-cond-list}, name\text{-alist}, proc\text{-list})$
 $\wedge \text{ok-mg-def-plistp} (proc\text{-list})$
 $\wedge \text{ok-mg-statep} (mg\text{-state}, r\text{-cond-list})$
 $\wedge (\text{code} (\text{translate-def-body} (\text{assoc} (subr, proc\text{-list}), proc\text{-list}))$
 $\quad = \text{append} (\text{code} (\text{translate} (cinfo, t\text{-cond-list}, stmt, proc\text{-list})),$
 $\quad \text{code2}))$
 $\wedge \text{user-defined-procp} (subr, proc\text{-list})$
 $\wedge \text{listp} (ctrl\text{-stk})$
 $\wedge \text{all-cars-unique} (mg\text{-alist} (mg\text{-state}))$
 $\wedge \text{signatures-match} (mg\text{-alist} (mg\text{-state}), name\text{-alist})$
 $\wedge \text{mg-vars-list-ok-in-p-state} (mg\text{-alist} (mg\text{-state}),$
 $\quad \text{bindings} (\text{top} (ctrl\text{-stk})),$
 $\quad temp\text{-stk})$
 $\wedge \text{no-p-aliasing} (\text{bindings} (\text{top} (ctrl\text{-stk})), mg\text{-alist} (mg\text{-state}))$
 $\wedge \text{normal} (mg\text{-state}))$
 $\rightarrow (\text{p} (\text{map-down} (mg\text{-state},$
 $\quad proc\text{-list},$
 $\quad ctrl\text{-stk},$
 $\quad temp\text{-stk},$
 $\quad \text{tag} (\text{'pc}, \text{cons} (subr, \text{length} (\text{code} (cinfo))))),$
 $\quad t\text{-cond-list}),$
 $\quad \text{clock} (stmt, proc\text{-list}, mg\text{-state}, n))$
 $= \text{p-state} (\text{tag} (\text{'pc},$
 $\quad \text{cons} (subr,$
 $\quad \text{if normal} (mg\text{-meaning-r} (stmt,$
 $\quad \quad proc\text{-list},$
 $\quad \quad mg\text{-state},$
 $\quad \quad n,$
 $\quad \quad \text{list} (\text{length} (temp\text{-stk}),$
 $\quad \quad \text{p-ctrl-stk-size} (ctrl\text{-stk}))))))$

```

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

- add-code, 2
- alist-element-type-conversion, 3
- all-cars-unique, 3, 4, 10, 11, 13, 15, 18, 24, 25, 28, 30, 31, 33, 34, 36, 38, 40, 42, 44, 46, 48, 51, 53, 54, 56, 57, 59, 61, 63, 65, 66, 69, 71
- array-identifier-nat-p-objectp, 4
- bindings, 6–72
- bool-literal-bool-objectp, 5
- bool-literal-bool-objectp2, 5
- boolean-identifierp, 3, 29, 52
- boolean-identifiers-have-boolean-literal-values, 3
- boolean-literalp, 3–5
- boolean-literalp-mapping, 4
- boolean-mg-to-p-simple-literal, 2
- boolean-mg-to-p-simple-literal2, 2
- call-actuals, 5–16, 19–26, 29–49, 52–67, 69
- call-name, 5–8, 10, 11, 13, 15, 17, 19–23, 25, 27, 29–31, 33, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52–55, 57, 59, 61, 63, 64, 66, 68, 71
- cc, 7–29, 31–45, 47–52, 54–62, 64–70, 72
- clock, 18, 28, 51, 71
- code, 6–72
- cond-subsetp, 17, 27
- definedp, 4, 5, 30, 53
- fetch-label, 16, 19, 26, 28, 47, 50, 51, 68, 70, 72
- find-label, 17, 19, 27, 29, 47, 50, 52, 68, 70, 72
- fix-small-integer, 5
- fix-small-integer-identity, 5
- ilessp, 2
- int-identifierp, 3
- int-identifiers-have-int-literal-values, 3
- int-literal-int-objectp, 5
- int-literalp, 2–5
- int-literalp-mapping, 4
- int-literalp-value-small, 5
- int-literals-mapping, 3
- integerp, 5
- label-alist, 2, 16, 19, 26, 28, 47, 50, 51, 68, 70, 72
- length, 3, 4, 6–72
- length-plistp, 3
- length-plistp-2-rewrite, 3
- lessp-add1-add1-2, 2
- lessp-add1-add1-add1-3, 2
- literal-type-conversion, 3
- map-down, 6, 18, 20, 28, 30, 51, 53, 71
- map-down-values, 6–17, 19–27, 29, 31–45, 47–50, 52, 54–60, 62–70, 72
- mg-alist, 4–72
- mg-alistp, 2, 3
- mg-bool, 2, 46
- mg-bool-ok-boolean, 46
- mg-cond-to-p-nat, 5, 7–17, 19–27, 29, 31–45, 47–50, 52, 54–62, 64–69, 71, 72
- mg-max-ctrl-stk-size, 7–17, 19–27, 29, 31–45, 47–50, 52, 54–62, 64–69, 71, 72
- mg-max-temp-stk-size, 3, 4, 7–17, 19–27, 29, 31–45, 47–50, 52, 54–69, 71, 72
- mg-meaning-predefined-proc-call, 1
- mg-meaning-r, 1, 16–19, 26–29, 47–52, 67, 68, 70–72

- mg-simple-constant-assignment-args-simple-identifierps, 19
- mg-simple-constant-assignment-exact-time-lemma, 27
- mg-simple-constant-assignment-step
 - 3, 20
 - 6, 23
 - 7, 25
 - s-1-2, 20
 - s-4-5, 22
- mg-simple-constant-eq-args-definedp, 53
- mg-simple-constant-eq-args-simple-identifierps, 52
- mg-simple-constant-eq-exact-time-lemma, 71
- mg-simple-constant-eq-step-10, 64
- mg-simple-constant-eq-step-11-f
 - alse-case, 68
- mg-simple-constant-eq-step-11-t
 - rue-case, 66
- mg-simple-constant-eq-step-4, 54
- mg-simple-constant-eq-step-5, 55
- mg-simple-constant-eq-step-6, 57
- mg-simple-constant-eq-step-7, 59
- mg-simple-constant-eq-step-8, 61
- mg-simple-constant-eq-step-9, 63
- mg-simple-constant-eq-steps-1-3, 53
- mg-simple-variable-assignment-args-definedp, 5
- mg-simple-variable-assignment-args-simple-identifierps, 6
- mg-simple-variable-assignment-exact-time-lemma, 17
- mg-simple-variable-assignment-step
 - 3, 7
 - 4, 8
 - 5, 10
 - 6, 11
 - 7, 13
 - 8, 15
 - s-1-2, 6
- mg-simple-variable-eq-args-definedp, 30
- mg-simple-variable-eq-args-simple-identifierps, 29
- mg-simple-variable-eq-exact-time-lemma, 50
- mg-simple-variable-eq-step-10, 42
- mg-simple-variable-eq-step-11, 44
- mg-simple-variable-eq-step-12-f
 - alse-case, 48
- mg-simple-variable-eq-step-12-t
 - rue-case, 46
- mg-simple-variable-eq-step-4, 31
- mg-simple-variable-eq-step-5, 32
- mg-simple-variable-eq-step-6, 34
- mg-simple-variable-eq-step-7, 36
- mg-simple-variable-eq-step-8, 38
- mg-simple-variable-eq-step-9, 40
- mg-simple-variable-eq-steps-1-3, 30
- mg-to-p-simple-literal, 2–5, 20–26, 42, 46, 48, 49, 54–67, 69
- mg-to-p-simple-literalp-preserve
 - s-untag-equality, 2
 - s-untag-ilessp, 2
- mg-vars-list-ok-in-p-state, 3, 4, 6–8, 10, 12, 13, 15, 18, 20–22, 24, 25, 28, 30, 31, 33, 35, 36, 38, 40, 42, 44, 46, 48, 51, 53, 54, 56, 57, 59, 61, 63, 65, 66, 69, 71
- mg-word-size, 4, 5, 7–17, 19–27, 29, 31–45, 47–50, 52, 54–69, 71, 72
- no-p-aliasing, 15, 18, 25, 28, 30, 31, 33, 35, 36, 38, 40, 43, 44, 46, 48, 51, 53, 54, 56, 57, 59, 61, 63, 65, 66, 69, 71
- normal, 1, 6–8, 10, 12, 13, 15, 16, 18, 20–22, 24–26, 28, 30, 31, 33, 35, 36, 38, 41, 43, 44, 46–49, 51, 53, 54, 56, 57, 59, 61, 63, 65, 67, 69–71
- ok-mg-def-plistp, 6–8, 10, 11, 13, 15, 17, 20–23, 25, 27, 30, 31,

33, 34, 36, 38, 40, 42, 44,
 46, 48, 51, 53, 54, 56, 57,
 59, 61, 63, 64, 66, 69, 71
 ok-mg-statement, 5–8, 10, 11, 13,
 15, 17, 19–23, 25, 27, 29–
 31, 33, 34, 36, 38, 40, 42,
 44, 46, 48, 50, 52–54, 56,
 57, 59, 61, 63, 64, 66, 69,
 71
 ok-mg-statep, 5–8, 10, 11, 13, 15,
 17, 19–23, 25, 27, 29–31,
 33, 34, 36, 38, 40, 42, 44,
 46, 48, 51–54, 56, 57, 59,
 61, 63, 65, 66, 69, 71
 ok-mg-valuep, 46
 ok-translation-parameters, 17, 27
 p, 18, 28, 51, 71
 p-ctrl-stk-size, 6–8, 10, 11, 13, 15–
 23, 25–31, 33, 34, 36, 38,
 40, 42, 44, 46–55, 57, 59,
 61, 63, 64, 66–68, 70–72
 p-frame, 8–12, 14, 16, 21–24, 26, 32–
 35, 37, 39, 41–43, 45, 46,
 49, 55–58, 60–67, 69
 p-objectp-type, 4, 5
 p-state, 7–17, 19–27, 29, 31–45, 47–
 50, 52, 54–69, 71, 72
 p-step, 6, 7, 9, 10, 12, 14, 16, 20, 21,
 23, 24, 26, 30, 32, 33, 35,
 37, 39, 41, 43, 45, 47, 49,
 53, 55, 56, 58, 60, 62, 64,
 65, 67, 69
 p-word-size, 4, 5
 plistp, 18, 28
 predefined-call-translation-2, 1
 predefined-proc-call-meaning-r-
 2, 1
 predefined-proc-call-sequence, 2
 push, 6–14, 16, 20–26, 31–46, 49,
 54–67, 69
 resource-errorp, 18, 28
 resources-inadequatep, 1, 6–8, 10,
 11, 13, 15, 17, 20–23, 25,
 27, 30, 31, 33, 34, 36, 38,
 40, 42, 44, 46, 48, 50, 53–
 55, 57, 59, 61, 63, 64, 66,
 68, 71
 rget, 11–14, 16, 36–41, 58, 60, 62
 rput, 15, 16, 25, 26, 45, 47, 49, 66,
 67, 69
 signal-system-error, 1
 signatures-match, 5, 6, 10, 12, 13,
 15, 18, 19, 24, 25, 28–31,
 33, 34, 36, 38, 40, 42, 44,
 46, 48, 51–54, 56, 57, 59,
 61, 63, 65, 66, 69, 71
 simple-identifier-mapping, 3
 simple-identifier-mapping-2, 4
 simple-identifier-mapping-3, 4
 simple-identifier-nat-p-objectp, 4
 simple-identifierp, 2–4, 6, 19, 29, 30,
 52
 simple-identifiers-have-simple-v-
 alues, 2
 simple-typed-literalp, 2, 3, 53
 simple-typed-literalp-mapping-li-
 stp, 2
 small-integerp, 3, 5
 small-integerp-mapping, 5
 small-naturalp, 3, 4
 special-conditions-mg-cond-to-p-
 -nat, 5
 tag, 2, 5–39, 41–47, 49–72
 top, 6–72
 translate, 1, 6–8, 10, 11, 13, 15–22,
 24–31, 33, 34, 36, 38, 40,
 42, 44, 46–54, 56, 57, 59,
 61, 63, 65–72
 translate-def-body, 6–8, 10, 11, 13,
 15, 17, 20–22, 24, 25, 27,
 30, 31, 33, 34, 36, 38, 40,
 42, 44, 46, 48, 51, 53, 54,

56, 57, 59, 61, 63, 65, 66,
 69, 71
 translate-proc-list, 6–17, 19–27, 29,
 31–45, 47–50, 52, 54–62, 64–
 70, 72
 type, 3, 4

 unlabel, 2
 unlabel-call, 2
 untag, 2–5, 11–14, 16, 25, 26, 36–
 42, 45–49, 58, 60, 62, 66,
 67, 69
 untag-boolean, 4
 untag-int-literal-integerp, 5
 user-defined-procp, 6–8, 10, 11, 13,
 15, 18, 20–22, 24, 25, 27,
 30, 31, 33, 34, 36, 38, 40,
 42, 44, 46, 48, 51, 53, 54,
 56, 57, 59, 61, 63, 65, 66,
 69, 71

 value, 3, 4, 6–16, 20–26, 31–47, 49,
 54–67, 69